

Model-Based Diversification for Sequential Exploratory Queries

Hina A. Khan¹ · Mohamed A Sharaf¹

Received: 15 November 2016/Revised: 9 March 2017/Accepted: 10 March 2017/Published online: 27 March 2017
© The Author(s) 2017. This article is an open access publication

Abstract Today, data exploration platforms are widely used to assist users in locating interesting objects within large volumes of scientific and business data. In those platforms, users try to make sense of the underlying data space by iteratively posing numerous queries over large databases. While *diversification of query results*, like other data summarization techniques, provides users with quick insights into the huge query answer space, it adds additional complexity to an already computationally expensive data exploration task. To address this challenge, in this paper we propose a diversification scheme that targets the problem of efficiently diversifying the results of multiple queries within and across different data exploratory sessions. Our proposed scheme relies on a model-based diversification method and an ordered cache. In particular, we employ an adaptive regression model to estimate the diversity of a diverse subset. Such estimation of diversity value allows us to select diverse results without scanning all the query results. In order to further expedite the diversification process, we propose an order-based caching scheme to leverage the overlap between sequence of data exploration queries. Our extensive experimental evaluation on both synthetic and real data sets shows the significant benefits provided by our scheme as compared to the existing methods.

Keywords Algorithms · Design · Diversification · Performance · Query processing

1 Introduction

Data exploration is a key ingredient in a widely diverse set of discovery-oriented applications including the ones from science, business and finance [1–3]. In those applications generally, users explore their data iteratively for locating interesting objects. This data discovery process is typically ad hoc and labour intensive as: (i) data sets are complex and heterogeneous, (ii) the users come from diverse backgrounds with different understanding of the underlying data sets, and (iii) the users may not have a prior notion of interesting objects. For instance, consider the Sloan Digital Sky Survey (SDSS) science database,¹ which describes over 140 million objects and is over 30 TB in size [2].

Both professional and amateur astronomers access the SDSS archive. The professional users typically pose fairly complex queries on position, colours and other attributes of sky objects, whereas the amateur users initially explore the data sets with simpler queries and as they learn more about the detailed properties of the stars and galaxies, they are expected to define more sophisticated queries. The most common queries against SDSS database are spatial queries involving a small region in the sky [4]. For instance, consider the following example:

Example 1 Assume an amateur user, who is curious about finding a region in the sky having group of objects with some interesting features. To locate those objects, the user needs to access the PhotoObj table in SDSS that has around 500 attributes, most of which are floating point numbers, and more than 600 million rows. Those attributes define various features of stars, galaxies and sky samples stored in SDSS. However, the user has no prior information about

✉ Hina A. Khan
h.khan3@uq.edu.au

¹ University of Queensland, Brisbane, QLD, Australia

¹ <http://www.sdss.org>.

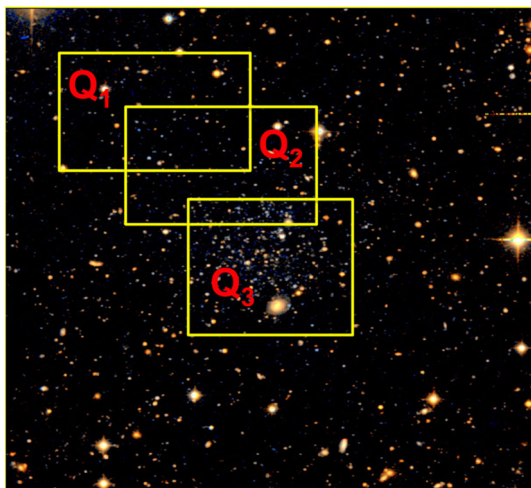


Fig. 1 Series of sequential queries generated in a data exploration session

which attributes and their corresponding values will make an object interesting. Hence, the user will start posing the initial queries with some intuitive coordinate values for specifying a certain area of the sky. As shown in Fig. 1, the user will first pose query Q_1 and review all the results returned by Q_1 . Guided by the results returned by Q_1 , the user may refine the query predicates to pose another query Q_2 . The result set of each query enhances the user's understanding of the underlying data and allow him to pose a more precise query. This iterative process continues until the user finally finds a region of interest, which is represented by query Q_3 in Fig. 1.

It should be clear from the Example 1 that the user interaction with the database takes the form of an *exploratory session* or *session* for short. During an exploration session, a user may execute multiple *sequential* queries that are initially vague and imprecise. Those queries potentially return thousands of results. It is essential for a user to review those results to assess their interestingness and refine their query based on them. However, comprehending large number of results is a time-consuming and challenging task. Therefore, to reduce the user effort and overall exploration time, data exploration platforms rely on several *representative data extraction techniques* to provide a synoptic understanding of the underlying query result. In particular, few representative results help user have a quick understanding of what is available in the whole result set and significantly reduce their time to review the results.

Thus, in the past years researchers have focused on developing effective representative data extraction methods. Beyond the well-studied top- k (e.g. [5, 6]), skyline (e.g. [7, 8]), sampling (e.g. [9, 10]) and clustering techniques (e.g. [11–13]), diversification is rapidly becoming

an important technique for extracting representative subsets with high coverage and minimum redundancy.

While diversification provides users with quick insights into the query answers and help them formulate their next query, it adds additional complexity and requires extra computational resources. In particular, the computational cost of diversifying the results of multiple sequential queries within an exploratory session increases linearly with the increase in the number of queries. Meanwhile, delivering near-real-time performance remains an essential requirement for data exploration platforms so that to match the intrinsic nature of iterative data exploration to ensure user satisfaction. Hence, motivated by the need to efficiently provide users with effective insights during data exploration we propose an efficient scheme for sequential diversification in data exploration platforms. Our proposed scheme, called *AdOr*, relies on two main interrelated components, namely: (1) an adaptive model-based diversification method and (2) an order-based caching scheme. In particular, *AdOr* employs an adaptive model-based diversification method to estimate the diversity of a diverse subset and hence selects diverse results without scanning all the query results. In order to further expedite the diversification process, *AdOr* employs an order-based caching scheme to leverage the overlap between sequence of data exploration queries.

In contrast to our previous work called DivIDE [14], the adaptive model employed by *AdOr* does not rely on the distribution of the underlying data or any prior model built for previous queries. Also in comparison to the random selection of cached results as presented in DivIDE, we prioritize the results in the cache according to their relevance to the partially computed diverse subset. Hence, leading to more efficient and effective selection of results from cache. Our extensive experimental evaluation on both real and static data sets shows the benefits achieved by *AdOr* in comparison to DivIDE and other diversification schemes.

In a nutshell the contributions of this paper are as follows:

- We formulate the problem of sequential diversification for iterative data exploration, together with metrics that capture both the efficiency and effectiveness of diversification (Sect. 2).
- We propose the novel *AdOr* scheme, which utilizes an adaptive model-based approach in contrast to a static model scheme, that is particularly suitable for the efficient and effective diversification in IDE (Sect. 4).
- We propose an ordered-based caching scheme that leverages the overlap between query results within and across exploratory sessions towards efficient diversification (Sect. 4.3.3).

- We conduct extensive experimental evaluation on real and synthetic data sets, which compare the performance of multiple diversification schemes and illustrate the benefits achieved by *AdOr* (Sect. 6).

Roadmap: We define the sequential search result diversification for data exploration platforms in Sect. 2. Next, we introduce the regression model for diversification in Sect. 3. In Sect. 4, we present the *AdOr* scheme. The evaluation test bed and results are reported in Sects. 5 and 6, respectively. We present related work in Sect. 7, and we conclude in Sect. 8.

2 Preliminaries

An *exploratory query* is typically based on range predicates represented as a multi-dimensional box or hyper-rectangle. Such queries retrieve a number of tuples, or objects, from the database, which can be generally viewed as a set of attribute values represented as data points in a multi-dimensional data space. It is essential for an IDE platform to help the user make quick decisions on the basis of few *representative* results, instead of sifting through all of those results looking for interesting insights. Search result diversification lends itself as an effective solution for extracting those representative results. Next, we present the preliminary concepts related to the Search Result Diversification problem. In Table 1, we summarize all the symbols and notations used in the rest of the paper.

2.1 Search Result Diversification

Formally defined, let $X = \{x_1, \dots, x_m\}$ be a set of results generated in response to some user query Q . In general, the goal of result diversification is to select a subset S^* of X

Table 1 Table of symbols

Symbol	Description
N	Total number of queries in a user session
k	Diverse subset size
Q_c	Current query to be diversified
X_c	Result set of Q_c
S_c	Diverse subset of X_c
x_{\max}	Result in X_c with maximum distance from S_c
\mathbb{Q}_H	Set of history queries
\mathbb{S}_H	Union of diverse subsets of history queries
$Q_{O,c}$	Set of queries overlapping with Q_c
$S_{O,c}$	Union of diverse subsets overlapping with X_c
$S_{R,c}$	Reusable diverse results for Q_c
θ	Deviation threshold
γ	Model threshold

with $|S^*| = k, k \leq m$, such that the diversity of the results in S^* is maximized.

There are various definitions of diversity [15]. Most of them can be classified in one of the following categories: (i) content based, i.e. selecting results that are dissimilar to each other (e.g. [15]) (ii) novelty based, i.e. selecting results that contain new information when compared to what was previously presented to the user (e.g. [16]) and (iii) semantic coverage based, i.e. selecting results that belong to different categories or topics (e.g. [17]).

In this work, we primarily focus on the widely used content-based definition of diversity. Content diversity is an instance of the *p-dispersion* problem [18], whose objective is to maximize the overall dissimilarity within a set of selected objects.

In particular, given a metric d that measures the distance between two results, e.g. the Euclidean distance among two data points, the diversity of a set S is measured by a *diversity function* $f(S, d)$ that captures the dissimilarity between the results in S . To that end, a number of different diversity functions have been employed in the literature, among which previous research has mostly focused on measuring diversity based on either the *average* or the *minimum* of the pairwise distances between results [19].

We focus on the second of those variants (i.e. minimum) known as max-min diversity measure, as it adopts a more uniform view to represent all the results in S , defined as:

$$f(S, d) = \min_{\substack{x_i, x_j \in X \\ x_i \neq x_j}} d(x_i, x_j)$$

where $d(x_i, x_j)$ represent the pairwise distance between any two results x_i and x_j in a multi-dimensional space which is typically measured under the L_p -norm metric. Without loss of generality, in this work $d(x_i, x_j)$ between two results in a D -dimensional space is calculated as Euclidian distance defined as:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^D (x_{i,m} - x_{j,m})^2}$$

Putting it all together, the max-min diversification problem is defined as follows:

Definition 1 Let X be the set of results that satisfy a user query Q and k be a positive integer such that $k \leq |X|$. Also, let d be a distance metric and f a diversity function. Then, max-min diversification is defined as selecting a subset S^* of X , such that:

$$\begin{aligned} S^* &= \operatorname{argmax} f(S, d) \\ S &\subseteq X \\ |S| &= k \end{aligned}$$

Identifying an optimal diverse subset S^* has been shown to be NP-hard (e.g. [20]); therefore, approximation methods are typically employed to select a near optimal diverse subset. Among many approximation methods, greedy-based heuristics are the ones most widely used because of their simplicity and performance. For instance, in *Greedy Construction* [19], the diversified set S is initialized by selecting a random result in X . Then, it proceeds through a number of iterations, until k results have been selected. In each iteration i , the result with the maximum set distance from the already selected results is added to the partially computed diverse set S_i .

From the above, it is straightforward to derive the complexity of Greedy Construction in terms of number of distance computation tasks as $O(k^2n)$ where $n = |X|$ [19]. However, caching distance computations across different iterations leads to further reduction in the number of pairwise distance computations, resulting in a complexity of $O(kn)$ [21].

2.2 Sequential Search Result Diversification

The series of queries submitted by a user during an exploratory session are typically correlated in a sense that the user formulates the next query in the sequence after having reviewed the results of previous queries [22]. This leads to an exploratory session, in which a user executes numerous selection queries iteratively using different predicates [23]. For instance, reconsider Example 1 where a user is searching for some interesting results in a two-dimensional space as shown in Fig. 1. The user will first pose query Q_1 and after reviewing the results she might reformulate it into another query Q_2 and so on. Since different queries within an exploratory session typically explore the data space in a close vicinity to each other, it is very likely for queries within a session to have overlapping results as shown in Fig. 1. Hence, the diversification of multiple related queries within a user session can be viewed as an instance of *continuous diversification* problem in which a diverse subset needs to be computed for each sliding window over dynamic data stream. Each exploratory query can also be perceived as a sliding window. However, in dynamic data stream a sliding window is moving along the time dimension, whereas an exploratory query is moving in space over static data. The result set of each query is generated from the new subspace of data that it covers, and a corresponding diverse subset of that result set needs to be computed. Hence, here we extend the definition of result diversification to the diversification of multiple sequential queries as defined below:

Definition 2 Let \mathbb{M} represent a set of multiple queries within an exploratory session. That is,

$\mathbb{M} = \{Q_1, Q_2, \dots, Q_N\}$. The sequential diversification problem is the following: For each query Q_i in \mathbb{M} with a result set X_i , compute a set S_i^* , such that:

$$\begin{aligned} S_i^* &= \operatorname{argmax}_{S_i \subseteq X_i} f(S_i, d) \\ |S_i| &= k \end{aligned}$$

where S_i is the subset of i th query result set and

$$f(S_i, d) = \min_{\substack{x_i, x_j \in S \\ x_i \neq x_j}} d(x_i, x_j)$$

Note that, in this work we focus on the max-min diversity measure to define $f(S_i, d)$, as it adopts a more uniform view to represent all the results in X_i .

The challenge of computing diverse subsets for each query Q_i in \mathbb{M} is that those diverse subsets cannot be computed incrementally. For instance, let X_i and X_{i+1} be result sets of two overlapping queries in \mathbb{M} . Then diverse subset S_{i+1} cannot be computed by updating the diverse subset S_i as the two diverse subsets may be completely different irrespective of the substantial overlap in the result sets X_i and X_{i+1} . Hence, each diverse subset S_i needs to be computed from scratch. Therefore, the computational cost of diversifying N queries in a session increases linearly with the increase in N . Hence, in this work we address the problem of designing efficient diversification methods that are scalable to N , i.e. length of a user session.

2.3 Problem Definition

To capture the performance of solutions for sequential diversification during a data exploration session, we define the following metrics:

Definition 3 Diversification cost, $C(S_i)$, is defined as the processing cost, in terms of number of distance and comparison operations, required to process a result X_i to produce a diversified result S_i of size k . Accordingly, the average diversification cost for a session of N queries is: $\frac{1}{N} \sum_{i=1}^N C(S_i)$.

Definition 4 Diversification quality, $D(S_i)$, is defined as the value of diversity $f(S_i, d)$ offered by the diversified result S_i . Accordingly, the average diversification quality for a session of N queries is: $\frac{1}{N} \sum_{i=1}^N D(S_i)$.

Specifically, our goal is to strike a fine balance between the efficiency of diversification (i.e. minimize $\frac{1}{N} \sum_{i=1}^N C(S_i)$) and its effectiveness (i.e. maximize $\frac{1}{N} \sum_{i=1}^N D(S_i)$). In order to achieve this goal our proposed

AdOr scheme utilizes an adaptive model to reduce the cost of diversification for each query in the exploratory session. *AdOr* further exploits the fact that the results of those queries are expected to exhibit some degree of overlap. Thus, *AdOr* leverages that overlap between query results by employing an order-based cache to store diverse results of previous queries. In the next section, we present our scheme and discuss its impact on both the efficiency and effectiveness of diversification.

3 Model-Based Diversification

As mentioned in Sect. 2.1, the Greedy algorithm iteratively computes a subset S of k diverse results from a query result set X . Specifically, in each iteration i of the Greedy algorithm, an optimal result $x_{\max}^i \in X/S$ is added to the partially computed diverse subset S^{i-1} such that:

$$f(S^{i-1} \cup x_{\max}^i, d) = \max(f(S^{i-1} \cup x_j, d) \mid x_j \in X/S)$$

The choice of next optimal result x_{\max}^i to be added to S^{i-1} is obviously made after examining all the candidate results in X . In this work, we take an alternate approach for identifying x_{\max}^i . Instead of examining all the results in X , we aim to predict the maximum value of the diversity function $f(S^i, d)$ that can be achieved by including another result to S^{i-1} in advance. This value can then be leveraged to *prune* the search space. Hence, if a result x_p that provides diversity value comparable to the estimated diversity value is found, then there is no need to search the query result set X any further. Thus, if x_p is the p th result in X then for remaining $|X| - p$ results, diversity function evaluation is not performed.

Clearly, several statistical and probabilistic models are applicable for estimation of diversity function. Such models have also been widely used for approximate query processing (e.g. [24, 25]). This includes models for Gaussian processes, interpolation, regression, dynamic probabilistic models, etc. In this work, we also adopt a *regression* model that is specifically suited to the diversification problem to efficiently and accurately estimate $f(S, d)$.

3.1 Regression Model for Diversity

Regression models provide a powerful tool for investigation of relationships between variables. The basic idea of regression models is to relate a dependent variable V_D to an independent variable V_I . Nonlinear regression, in particular, is a good choice when there is an evidence to believe that the relationship between the dependent variable and

independent variable follows a particular functional form [26]. A nonlinear regression model has the form:

$$V_D = f'(V_I, \beta) + \epsilon$$

where V_D is dependent variable, f' is a known function of the independent variable V_I including parameters defined by β , and ϵ is a random error [26].

In the case of diversification, the diversity value of a diverse set of results S clearly depends upon the number of results in that set (i.e. k). Specifically, small diverse subsets tend to exhibit higher diversity, whereas the value of diversity decreases with increasing the size of S . Hence, to ascertain the causal effect of diverse subset size k upon the value of the diversity function $f(S, d)$, we assume the value of the diversity function to be the dependent variable, where the diverse subset size is the independent variable.

To formally model the dependency between $f(S, d)$ and k , we plot a *diversity curve* as shown in Fig. 3 for a sample query result generated over a uniform data set (for more details on the different data sets we have experimented with, please see Sect. 5). The curve is generated by plotting the diverse subset size k on the x-axis and the diversity function value for the corresponding diverse subset on the y-axis.

As Fig. 3 shows, the diversity curve clearly exhibits a diminishing marginal gain trend. This is due to the fact that as new results are added using Greedy, the set of similar results already selected increases. Thus, with each new addition to the diverse subset the marginal gain in diversity decreases, which is consistent with the study done in previous work [27]. Such trend allows us to use a simple model like power series to model the diversity curve (Fig. 3).

Hence, we can alternatively represent the diversity function $f(S, d)$ in terms of a computationally inexpensive hypothesis function $f'(k)$ such that:

$$\text{Hypothesis Function } f'(k) = ak^{-b}$$

where a and b are the parameters we seek that would best fit the function to the sample data. Standard statistical calculations are used to determine the values of parameters a and b . Particularly, we use least squares estimation to measure the fitness of the regression model and root mean square error to measure the accuracy of predicted values. Other estimations can also be considered to accommodate the impact of outliers in data. The details of how parameters a and b are evaluated are given in Appendix A.

As mentioned above, the regression model relies on the observational data to evaluate the hypothesis function parameters. However, in case of query diversification, those observational values are only available once the query result has been diversified. Hence, in order to learn the regression parameters we consider two possible approaches. First, we present a baseline static approach that

we proposed previously in [14] and then we generalize it to an adaptive approach.

3.1.1 Static Model Approach

The static model approach relies on the observations made using a sample query over the database. These observations are then used to evaluate the model parameters a and b in the hypothesis function. Once those parameters are known, they remain static across various user queries. For instance, to learn the regression model parameters, a global sample query Q_G is executed over the database D to retrieve sample result set X_G . The Greedy algorithm is then applied over X_G to select a diverse subset S_G of size k . In each iteration i , where $i \leq k$, the diversity function value $f(S_G^i, d)$ is evaluated and the pair $(i, f(S_G^i, d))$ is plotted. As Greedy finishes execution, we have $k-1$ observations in the form of $(i, f(S^i, d))$ pairs. These observations are used for evaluating values of parameters a and b . Since those values are based on observations generated by a single global query Q_G , they depend upon: (1) the size of the data subspace accessed by Q_G and (2) the distribution of data within that subspace. The hypothesis function parameters need to be scaled for queries accessing data subspaces of different sizes. The parameter b , called either the exponent or the power, determines the rate of decay of our hypothesis function and the overall shape and behaviour. However, the parameter a serves as a simple scaling factor, moving the values of k^{-b} up or down as a increases or decreases, respectively. Since the shape of the diversity curve remains consistent for different data subspaces, only parameter a is scaled for every user query Q by a ratio of the data space covered by Q as compared to the space covered by Q_G .

3.1.2 Adaptive Model Approach

As discussed above, the static model approach relies on the observation data generated using a sample global query Q_G . The regression model built using query Q_G can be scaled to be used for each user query. However, the scaling of model parameters is based on the assumption that data distribution is uniform across all subspaces. For instance, in a clustered data set, the distribution can vary to a great extent for different subspaces. In particular, some of the queries can have overly populated result set while others return only few results even if the size of the data space accessed is the same. In such scenarios where distribution of underlying data is not uniform, the static model scheme seems to be inadequate.

For instance, as shown in Fig. 4, the size of data space covered by Q_1 is the same as the space covered by Q_2 ; however, the data distribution for both queries is very different. Although the diversity values evaluated for both queries, for the same values of k , are expected to be quite different, scaling global regression model for both Q_1 and Q_2 will generate same values for parameters a and b . For example, Fig. 5 shows the corresponding diversity curves for both Q_1 and Q_2 . Clearly, the scaled regression model fits the diversity curve of Q_1 while fails to adjust to the diversity curve of Q_2 .

To address the limitations of static model approach, we consider a more general approach based on adaptive regression model that learns model parameters individually for each user query and does not rely on any prior observations or distribution of underlying data. The main idea underlying the adaptive model approach is to utilize the observations from the current query being diversified. Instead of using an existing global model, for each individual query the regression model is built on the go as we diversify the query. The adaptive model approach can be summarized in the following steps:

- Step 1: An initial diverse subset S of size two is generated using Greedy Construction algorithm.
- Step 2: The diversity function value of S is used as first observation and initial values for model parameters a and b are evaluated.
- Step 3: Based on the hypothesis function, diversity function value is estimated as Div_e for the next iteration of Greedy algorithm.
- Step 4: New result with the maximum set distance from the already selected results in S is added to S . The actual diversity function value for newly updated set S is computed as Div_a and added to the observation data.
- Step 5: The difference in the actual diversity function value (Div_a) and the estimated diversity value predicted using hypothesis function (Div_e) is computed. The regression model is assumed to be stable if the following condition is true:

$$\gamma \geq \frac{Div_a - Div_e}{Div_e}$$

where γ is the threshold value defining the acceptable difference ratio between estimated diversity value and the actual diversity value. If the above condition is not met, then the actual diversity value Div_a is added to the observation data.

- Step 6: The new values of model parameters a and b are computed based on updated observations, and Steps 3 to 5 are repeated.

4 AdOr Scheme for Sequential Diversification of Multiple Queries

In this section, we present our *AdOr* scheme for the efficient diversification of query results in data exploration platforms. The main idea underlying *AdOr* is to leverage an adaptive regression model to estimate the diversity of a subset and utilize the overlap in a sequence of queries (such as the one shown in Fig. 1) in order to minimize the computational cost incurred in diversifying each query result set. Towards that goal, AdOr exploits two novel techniques, namely: (1) regression model-based Greedy algorithm and (2) caching of diverse results. Before going through the details of each of those techniques, we first present our baseline solution for the diversification of multiple sequential queries, namely: Greedy Construction.

4.1 Greedy Construction

In this baseline solution, the *Greedy Construction* algorithm (or *Greedy* for short), is directly applied for the diversification of results. In particular, Greedy (as presented in Algorithm 1) evaluates the diverse subset S of each query iteratively by computing set distances of all the

results in a query result set X . For instance, in order to select a new result in partially computed diverse subset S_i , Greedy computes the set distance of each result in X from results already selected in S as:

$$setDist(x, S) = \min_{\substack{x \in X \\ x_j \in S}} d(x, x_j)$$

The result x_{max} having the maximum set distance is added to S . Thus, in k iterations Greedy selects k results to be added to diverse subset S . Also, Greedy processes each query result individually without taking into consideration the overlapping in results between different queries.

Recall from Sect. 2.1, the computational complexity of diversifying a single query using Greedy algorithm is $O(k^2|X_i|)$, where $|X_i|$ is the size of the query result and k is the size of the diverse subset. Clearly, applying Greedy algorithm independently to each exploration query results in a computational complexity that increases linearly with the increase in the number of total queries in a user session (i.e. N). That is, the complexity of Greedy algorithm for N queries is simply computed as: $O(k^2|X_1|) + \dots + O(k^2|X_N|)$.

Algorithm 1 Greedy.

Input: A set of query results X , an integer k .
Output: A set S^k with the k most diverse results in X .

```

1:  $x \leftarrow$  random result  $\in X$ 
2:  $i \leftarrow 1$ 
3:  $S^i \leftarrow \{x\}$ 
4:  $i \leftarrow i + 1$ 
5: while  $i < k$  do
6:    $x_{max} \leftarrow \operatorname{argmax}_{x_{max} \in X} \min_{x_j \in S^{i-1}} d(x_{max}, x_j)$ 
7:    $S^i \leftarrow S^{i-1} \cup \{x_{max}\}$ 
8:    $i \leftarrow i + 1$ 
9: end while
10: return  $S^k$ 

```

Algorithm 2 mGreedy

Input: Query Result set $X = (x_1, x_2, \dots, x_m)$, Set of previous overlapping diverse results S_R , deviation threshold θ , a hypothesis function $f'(i)$, Partially computed diverse subset S , Diverse subset size k

Output: S with k diverse results

```

1:  $X' \leftarrow X \setminus S_R$ 
2: while  $|S^i| < k$  do
3:   if (Best Fit) then
4:      $\bar{x}_{max} \leftarrow \operatorname{argmax}_{\bar{x}_{max} \in S_{R,c}} \min_{x_j \in S^{i-1}} d(\bar{x}_{max}, x_j)$ 
5:      $S^i \leftarrow S^{i-1} \cup \{\bar{x}_{max}\}$ 
6:     if ( $\operatorname{deviation}(\bar{x}_{max}) > \theta$ ) then
7:        $S^i \leftarrow S^{i-1} - \{\bar{x}_{max}\}$ 
8:        $x_{max} \leftarrow \operatorname{argmax}_{x_{max} \in X'_c} \min_{x_j \in S^{i-1}} d(x_{max}, x_j)$ 
9:     end if
10:  end if
11:  if (First Fit) then
12:     $\bar{x}_{max} \leftarrow x_j$  s.t  $x_j \in S_{R,c}$ ,  $\operatorname{deviation}(x_j) \leq \theta$ 
13:    if ( $\bar{x}_{max} = \phi$ ) then
14:       $\bar{x}_{max} \leftarrow x_j$  s.t  $x_j \in X'_c$ ,  $\operatorname{deviation}(x_j) \leq \theta$ 
15:    end if
16:     $x_{max} \leftarrow \bar{x}_{max}$ 
17:  end if
18:   $S \leftarrow S \cup x_{max}$ 
19: end while
20: return  $S$ 

```

4.2 Model-based Greedy Algorithm

In this section, we present a model-based Greedy algorithm called *mGreedy* that selects the diverse query results using the prediction of a regression model. We assume that the hypothesis function $f'(k) = ak^{-b}$ has been defined with parameters a and b using either static or adaptive approach as detailed in Sect. 3.1. Note that, in case of adaptive approach some of the diverse results are already selected and *mGreedy* is applied to select the remaining diverse results. Let Q_c be the current query and X_c be the respective result set of Q_c . Like, Greedy algorithm, *mGreedy* also builds the diverse subset iteratively. In each iteration i , an estimated value of the diversity function is computed using the hypothesis function formulated above (i.e. $f'(i)$). To decide on the next result to be added to S_c^{i-1} , *mGreedy* computes a *deviation* value of each result $x \in X_c$ with respect to the diverse subset S_c^i as:

$$\operatorname{deviation}(x) = |f'(i) - f(S_c^{i-1} \cup \{x\}, d)|$$

Particularly, *mGreedy* selects a result \bar{x}_{max} , which has the deviation value less than a user-specified threshold value, θ . As soon as such a result is identified the iteration is terminated. Hence, if \bar{x}_{max}^i is the p th result in X_c then this saves $|X_c| - p$ set distance calculations for the remaining candidate results in X_c . It is important to mention here that \bar{x}_{max}^i is an approximation of x_{max}^i . Thus, instead of adding x_{max}^i to S_c^{i-1} the approximated result \bar{x}_{max}^i is added to S_c^{i-1} . Therefore, $f(S_c^{i-1} \cup \{x_{max}^i\}, d) \geq f(S_c^{i-1} \cup \{\bar{x}_{max}^i\}, d)$.

Applying *mGreedy* individually to each query reduces the cost of diversification; however, in the worst-case

scenario it might have to look at large number of results before it locates \bar{x}_{max}^i . In order to locate the promising results quickly, we make use of the overlapping diverse results from the previous queries. In the next section we discuss how AdOr exploits caching to leverage the overlap between different queries and reduces the cost of diversification even further.

4.3 Cache-based Sequential Diversification of Overlapping Queries

A data exploration session generally involves multiple related queries, and overlapping between results of those queries is naturally expected to occur. To formally express that overlap between exploratory queries, consider a current query Q_c for which a diversified set S_c is to be computed, and a history of processed queries $\mathbb{Q}_H = \{Q_1, Q_2, \dots, Q_{c-1}\}$. Hence, there exists $\mathbb{Q}_{O,c} \subseteq \mathbb{Q}_H$ such that the result X_i of each query $Q_i \in \mathbb{Q}_{O,c}$ overlaps with the result X_c of the current query Q_c (i.e. $X_i \cap X_c \neq \phi$).

Clearly, the diversified results of these overlapping queries $\mathbb{Q}_{O,c}$ can be utilized for reducing the cost of diversifying Q_c . Particularly, in this work, we propose using a *cache* of diversified results for improving the efficiency of diversification. A Cache \mathbb{S}_H , which contains the diversified results of all previously processed queries \mathbb{Q}_H , is expressed as follows:

Definition 5 Cached diverse results, \mathbb{S}_H , is the set of diversified results corresponding to the queries in \mathbb{Q}_H . That is, $\mathbb{S}_H = \{S_1, S_2, \dots, S_{c-1}\}$.

Given a query Q_c , it is then straightforward to fetch the cached diverse results of the set of queries $\mathbb{Q}_{O,c}$ that overlap with Q_c . We denote the cached diverse results of queries overlapping with Q_c as $S_{O,c}$, which is defined as:

Definition 6 Diverse results of overlapping queries, $S_{O,c}$, is the union of the diversified sets of all the queries in $\mathbb{Q}_{O,c}$.

Intuitively, $S_{O,c}$ is expected to contain some points that are common with the results of query Q_c (i.e. $X_c \cap S_{O,c} \neq \phi$). That set of common points $S_{R,c}$ provides an excellent opportunity to *reuse* in constructing the diverse set S_c and is simply defined as follows:

Definition 7 Reusable diverse results, $S_{R,c}$, such that $S_{R,c} \subseteq S_{O,c}$ and contains all diverse results in $S_{O,c}$ that fall in the range of Q_c (i.e. $S_{O,c} \cap X_c$).

Thus, one idea towards utilizing overlapping result set $S_{R,c}$ across multiple queries when diversifying Q_c is to simply initialize S_c with $S_{R,c}$. Clearly, however, that idea has a major drawback that is: the results in $S_{R,c}$ might exhibit high degree of redundancy. This will lead to a selection of many non diverse results in S_c . To further explain that point, notice that the results in each $S_i \in S_{O,c}$ are diverse in their own. However, this assumption breaks once combining some of those diverse results together into $S_{R,c}$. This is because the results in two different sets S_i and S_j might be very similar to each other if their corresponding queries explored roughly the same data subspace.

For instance, Fig. 6 shows result sets of three overlapping queries. As shown in figure, X_c is the result of a newly submitted query Q_c , which overlaps with two other historical results X_1 and X_2 . The diversified set S_1 extracted from X_1 is shown in squares, whereas the diversified set S_2 extracted from X_2 is shown in triangles. The diverse results from both queries that overlap with X_c are retrieved and form the set of

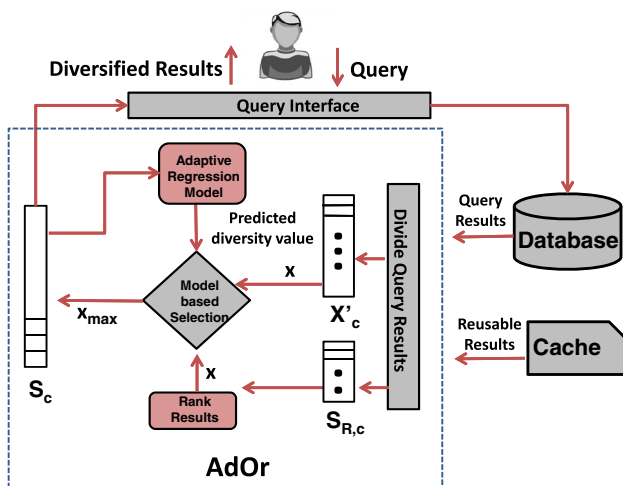


Fig. 2 AdOr architecture

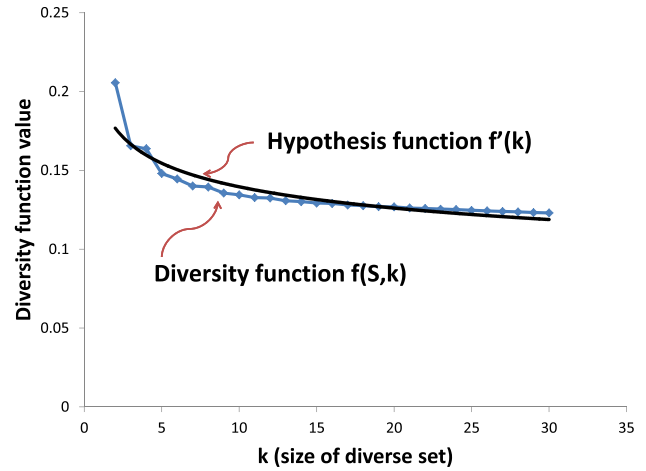


Fig. 3 Mapping of diversity function to hypothesis function using regression modelling

reusable diverse results $S_{R,c}$. It is clear from Fig. 6, however, that set $S_{R,c}$ may contain few results that are very similar to each other. Thus, initializing diverse subset S_c with $S_{R,c}$ will adversely affect the final diversity of set S_c .

Taking advantage of natural overlap occurring between various queries during an exploration session and at the same time ensuring that only dissimilar results are selected to be included in the diverse subset of current query is precisely the goal of our AdOr scheme. Next we present in detail how the AdOr scheme achieves this goal.

4.3.1 Dividing the Search Space

Recall that in each iteration of the mGreedy algorithm, the first result that provides the diversity close to the estimated diversity value as predicted by the regression model is added to the diverse subset. Clearly, with random order of results, the optimal result may reside in the end of the result

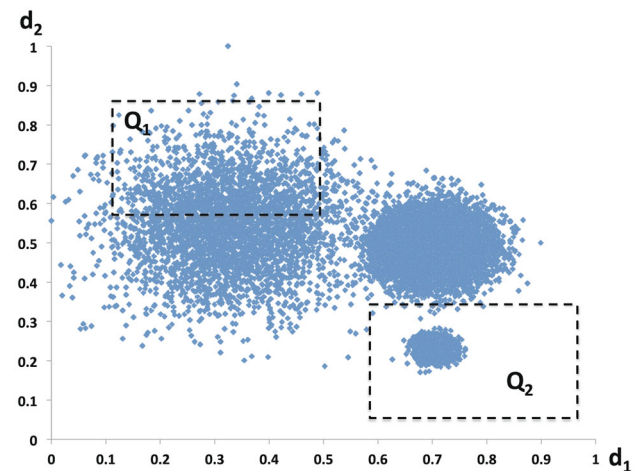


Fig. 4 Model-based diversification for clustered data

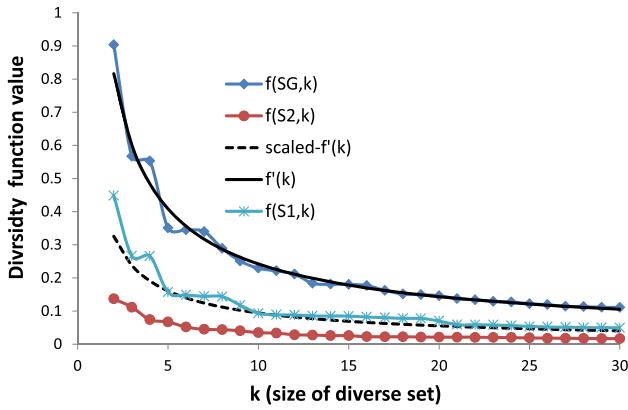


Fig. 5 Scaling hypothesis diversity function for different queries

list requiring many set distance computations for results appearing before it. For large data sets where numerous results are returned in response to each query, and where near-real-time performance is expected, the goal of AdOr is to minimize the cost incurred in locating the result \bar{x}_{\max}^i in potentially large search space defined by the results in X_c . To minimize that cost, AdOr *divides* the search space (i.e. X_c) into two disjoint subsets, as follows:

1. *Reusable Diverse Results ($S_{R,c}$):* The set of diversified results of $\mathbb{Q}_{O,c}$ that overlap with the results in X_c . That is, as defined above, $S_{R,c} = X_c \cap S_{O,c}$.
2. *New Query Results (X'_c):* This is the set of results in X_c after excluding $S_{R,c}$. That is, $X'_c = X_c \setminus S_{R,c}$.

4.3.2 Selection of Diverse Results Using Cache

As mentioned above, the set of results in $S_{R,c}$ have high potential to appear in the diversified set S_c . However, instead of blindly initializing S_c with $S_{R,c}$, AdOr alternates between the two sets of results (i.e. X'_c and $S_{R,c}$) in an efficient manner and during that process it “selectively” chooses only the most promising results guided by the regression model so that to avoid compromising the quality of diversification.

As shown in Fig. 2, to diversify a result set X_c , AdOr divides X_c into two subsets: the reusable diverse result set $S_{R,c}$ and the new query result set X'_c . For selecting that result \bar{x}_{\max}^i , AdOr uses the following two alternative selection methods:

- *First-Fit Diversification (AdOr-FF):* Search the set $S_{R,c}$ for the first result \bar{x}_{\max}^i , which has the deviation value less than the deviation threshold θ . If such a result is found, it is added to set S_c^{i-1} , else X'_c is searched until \bar{x}_{\max}^i is located.
- *Best-Fit Diversification (AdOr-BF):* Search the set $S_{R,c}$ for the result \bar{x}_{\max}^i , which will maximize the diversity function value if added to set S_c^{i-1} . If the *deviation*(\bar{x}_{\max}^i) is less than deviation threshold θ , then

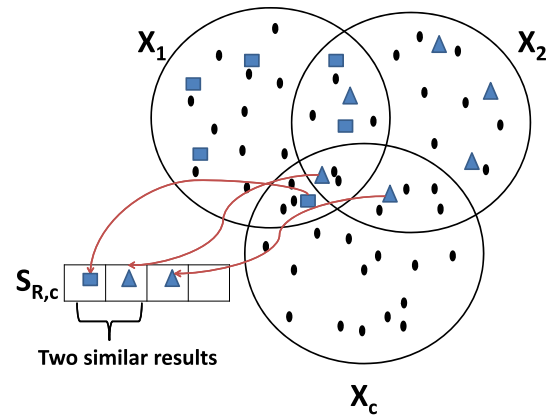


Fig. 6 Reusable Set $S_{R,c}$ for current query result set X_c generated from diverse overlapping results of previous queries

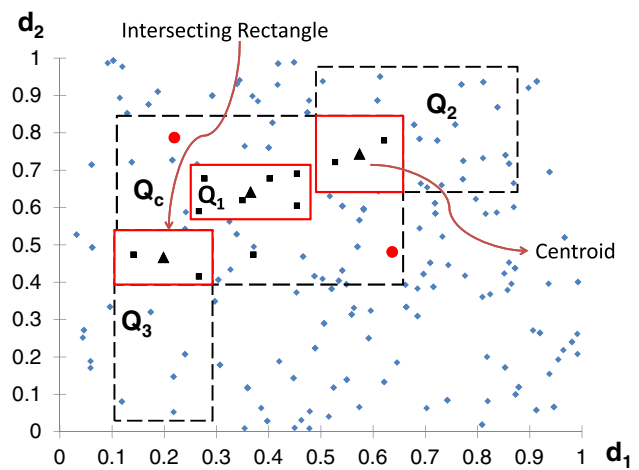


Fig. 7 Ordering of overlapping cached diverse results

\bar{x}_{\max}^i is added to the set S_c^{i-1} . Otherwise, the optimal result x_{\max}^i is located in X'_c and in turn, added to S_c^{i-1} .

Clearly, each time a result is located in $S_{R,c}$, it saves at least $|X_c| - |S_{R,c}|$ number of distance calculations and data comparison operations. In particular, AdOr-FF picks the first result that provides comparable diversity to the one predicted by the regression model, whereas while searching $S_{R,c}$, AdOr-BF evaluates the deviation value of the result with maximum set distance from the diverse subset S_c^{i-1} . In order to locate that result AdOr-BF examines all the results in $S_{R,c}$. Thus, AdOr-BF tries to find diverse subsets with higher diversity at the expense of higher computational cost.

4.3.3 Ordering of cached diverse results

Obviously, the performance of AdOr-FF depends a lot on the order in which the results are stored in $S_{R,c}$. If the results distant from the current partial diverse subset $S_c^i - 1$ appear earlier in the search list, then the cost savings are

higher. Keeping this goal in mind, we propose an ordering scheme that processes the promising overlapping diverse subsets first. In order to elaborate the ordering scheme consider the following example.

Let Q_c be the current query with result set X_c for which a diverse subset S_c needs to be generated. Also, let $\mathbb{Q}_{O,c}$ be set of three previous queries overlapping with Q_c containing Q_1, Q_2 and Q_3 as shown in Fig. 7. In a random iteration i of greedy heuristic the diverse results already selected in S_c^i are shown in red circle shapes. The overlapping diverse results from the previous queries are shown in square shapes. As clear from Fig. 7, the results in Q_3 are most distant from the results already in S_c^i . However, since the results are stored in the order in which the queries were generated, the results in Q_3 will be examined last.

Thus, in order to prioritize the order in which queries from $\mathbb{Q}_{O,c}$ are examined, we identify the intersecting rectangles between Q_c and each of the overlapping cached queries in $\mathbb{Q}_{O,c}$ as shown in solid boundaries in Fig. 7. Next, we determine the centroid of each of these intersecting rectangles. Those centroids represent their respective query in $\mathbb{Q}_{O,c}$ and are shown as diamond shapes in Fig. 7. The set distance between each query Q in $\mathbb{Q}_{O,c}$ from the diverse subset S_c^i is calculated as:

$$setDist(Q, S_c^i) = setDist(centroid_Q, S_c^i)$$

The queries with higher set distance from the diverse subset S_c^i have the higher potential of containing \bar{x}_{max}^i as

compared to the queries with smaller set distances from S_c^i . Hence, each query in $\mathbb{Q}_{O,c}$ is assigned a priority score as: $Score^i(Q) = setDist(Q, S_c^i)$. All the queries in $\mathbb{Q}_{O,c}$ are then processed in the decreasing order of their scores. It means the diverse results from the most distant overlapping query are processed first.

It should be noted that $Score^i(Q)$ is the priority of Q with respect to S_c^i in iteration i . As the diverse subset S_c evolves in subsequent iterations the score of each query in $\mathbb{Q}_{O,c}$ also changes. Hence, in each iteration the score of each overlapping query is re-evaluated by computing a set distance between the centroid of the query and the diverse subset S_c .

The additional cost of computing priority scores in terms of number of set distance computations is equal to the number of overlapping queries ($|\mathbb{Q}_{O,c}|$). Since the number of overlapping queries is usually much small as compared to the number of reusable diverse results ($|\mathbb{Q}_{O,c}| \ll |S_{R,c}|$), overall cost savings in terms of number of distance computations can still be expected. However, with increasing number of queries in cache the number of queries overlapping with the Q_c can also increase. This will not only have an impact on the computational cost of priority scores but will also increase the storage cost. Hence, in order to keep the number of cached queries limited yet achieving the benefits of previously cached results we employ some effective cache management techniques as discussed next.

Algorithm 3 AdoR

Input: Query Result set $X_c = (x_1, x_2, \dots, x_m)$, an integer k , Set of previous overlapping diverse results $S_{R,c}$, deviation threshold θ , a hypothesis function $f'(i)$

Output: Set $S^k = (s_1, s_2, \dots, s_k)$ with the k most diverse items of X_c .

```

1: prediction ← false
2: i ← 1
3: x ← a random result in  $X_c$ 
4:  $S^i \leftarrow \{x\}$ 
5:  $i \leftarrow i + 1$ 
6: if  $f'(i) = \text{Null}$  then
7:    $x_{max} \leftarrow \text{argmax}_{x_{max} \in X_c} \min_{x_j \in S^{i-1}} d(x_{max}, x_j)$ 
8:    $S^i \leftarrow \{x_{max}\}$ 
9:    $O \leftarrow O \cup \{(i, f(S^i, d))\}$ 
10:   $f'(i) \leftarrow \text{BuildModel}(O)$ 
11:   $i \leftarrow i + 1$ 
12: while NOT (StableModel) do
13:    $estdiv \leftarrow f'(i)$ 
14:    $x_{max} \leftarrow \text{argmax}_{x_{max} \in X_c} \min_{x_j \in S^{i-1}} d(x_{max}, x_j)$ 
15:    $S^i \leftarrow \{x_{max}\}$ 
16:   if  $(\text{Ratio}(estdiv, f(S^i, d)) \leq \gamma)$  then
17:     StableModel ← true
18:   else
19:      $O \leftarrow O \cup \{(i, f(S^i, d))\}$ 
20:      $f'(i) \leftarrow \text{BuildModel}(O)$ 
21:   end if
22:    $i \leftarrow i + 1$ 
23: end while
24: end if
25:  $S^k \leftarrow S^i \cup m\text{Greedy}(X_c, S_{R,c}, S^i, \theta, f'(i), k - i)$ 
26: return  $S^k$ 

```

4.4 Cache Management

In an interactive exploration environment where multiple queries are generated within and across user sessions, it is very likely that after a while the size of the cache (i.e. S_H) will grow to match the size of underlying data set. This will pose two challenges:

- As the size of cache increases, the search time for locating a reusable set $S_{R,c}$ for a query Q_c also increases.
- The size of the reusable set $S_{R,c}$ may grow to become larger than the size of X'_c .
- The computational cost of evaluating scores for ordering overlapping queries may outweigh the benefits of ordering.

As a consequence of the observations above, a large cache size would reduce the amounts of savings provided by AdOr until it reaches zero. That is, the processing cost of AdOr becomes similar to that of Greedy. The obvious approach to address these challenges is to limit the reusable set $S_{R,c}$ size. That is, instead of using the entire reusable set $S_{R,c}$, a subset $S'_{R,c}$ is used. Accordingly, as new queries are posed, some of the diverse subsets are evicted from cache to make room for new subsets. Hence, when the number of diverse subsets become greater than L , AdOr replaces one of the stored diverse subset.

Clearly, choosing the best cache replacement policy is application dependent and can be determined on the basis of query trends in a particular database application. In this work, we simply adopted the popular *least frequently used (LFU)* cache replacement policy to evict the diverse subset that is accessed the least number of times.

5 Experimental Test bed

We perform a number of experiments to evaluate the efficiency and the effectiveness of our AdOr scheme. In particular, we compare AdOr against two baseline approaches: Greedy and static model approach. Table 2 summarizes the different parameters used in our experimental evaluation.

Schemes: We evaluate the performance of the following schemes:

- *Greedy:* Applies the Greedy Construction heuristic independently on each query result set to select the respective diverse subset.
- *SGC:* Uses Stream Greedy Construction heuristic as presented in [28]. SGC relies on the overlap that occurs between the results of two consecutive queries (i.e. $X_i \cap X_{i-1}$). The diverse subset S_i is initialized with the r

Table 2 Evaluation setting

Parameter	Range	Default
Number of queries (N)	2–1000	100
Diverse subset size (k)	10–40	30
Number of cached queries (L)	20–100	20
Deviation threshold (θ)	–	0.05
Model threshold (γ)	0.01–0.05	0.02
Data size (D)	20–40k	20k
Data sets	Unif., Clust., SDSS	Unif.

overlapping diverse results from S_{i-1} . Thus, only $k - r$ remaining diverse results are computed using Greedy heuristic. The details of SGC scheme are given in Sect. 7.

- *Static-No-Cache:* Uses static model approach to build the regression model based on sample observations generated by a global query. The regression model is then used to predict future values of diversity function for the selection of diverse subsets across various queries.
- *Adaptive-No-Cache:* Applies adaptive model approach to build a regression model for each individual query.
- *Adaptive-Random-Cache:* Extends adaptive model approach to use diverse results of previous overlapping queries stored in cache. The diverse results from cache are accessed in random order.
- *Adaptive-Ordered-Cache:* Employs a priority scheme to order the results of overlapping queries in cache. The diverse results in cache are accessed in decreasing order of their distance from already selected diverse results for the current query.

For all the schemes using cached overlapping diverse results, we further evaluate both best-fit and first-fit alternatives (Sect. 4.3.2).

Performance Measures: The performance of each algorithm is measured based on the following metrics:

- Cost ($\sum_{i=1}^N C(S_i)$) measured as the sum of operations performed to evaluate diverse subsets of N queries, where each operation represents a distance computation and a comparison evaluation.
- Diversity ($\frac{1}{N} \sum_{i=1}^N D(S_i)$) measured as average diversity across the diversified subsets of N queries.

Data sets: We use both synthetic and real data sets. Our synthetic data sets consist of points in the two-dimensional Euclidean space. Points are either uniformly distributed (“Uniform”) or form clusters around a random number of points (“Clustered”). Our real data set is based on the SDSS database and contains 40k data rows. We use the uniformly distributed numerical columns `rowc` and `colc`

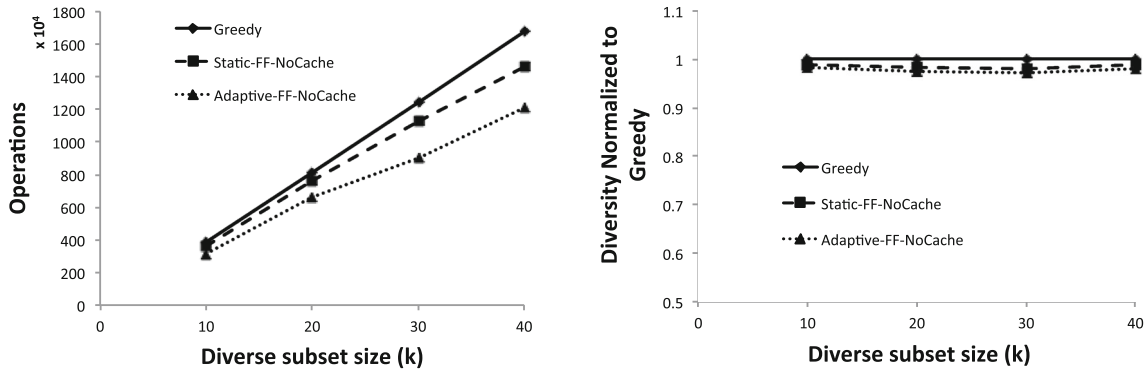


Fig. 8 Adaptive model versus static model

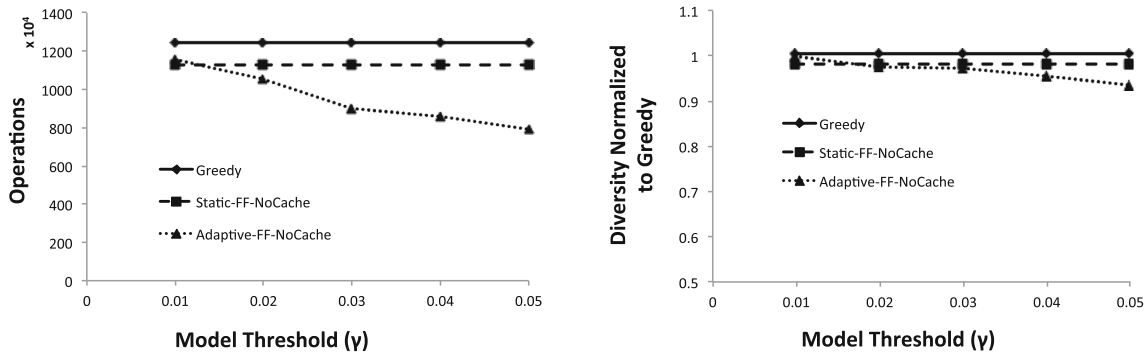


Fig. 9 Impact of γ on cost and diversity

from PhotoObjAll table. For all data sets, attribute values are normalized to [0–1].

Queries: We simulate random user sessions with multiple range queries. For each experiment, the number of total queries N across sessions is in the range [2–1000]. Each query is also associated with the size of diverse set k , which takes values in the range [10–40]. Cache is initialized with diverse results of 20 queries.

6 Experimental Evaluation

In the following experimental results, we evaluate the sensitivity of AdOr to the different parameters discussed in the previous section.

6.1 Impact of Adaptive Regression Model

In this experiment, we compare the performance of adaptive model approach against the static model approach. All the queries are generated over clustered data set to evaluate the proficiency of adaptive regression model. We compare the performance of both schemes without using any cached results to emphasize on the impact of model-based diversification alone. Hence, under

this experimental setting the performance of best-fit diversification approach is similar to Greedy. Therefore, we compare the performance of only first-fit alternatives by varying the following parameters.

6.1.1 Impact of varying Diverse Set Size

As shown in Fig. 8b, both adaptive and static model schemes perform less number of operations as compared to Greedy. However, cost of the static model scheme is upto 15% higher as compared to adaptive model. This is because the static model fails to adjust for queries with different data distributions, and hence, the diverse results are located by examining most of the query results. Figure 8a shows that both schemes locate diverse subsets with diversity comparable to the subsets located by Greedy heuristic.

6.1.2 Impact of varying γ

In this experiment, we focus on the impact of varying threshold parameter γ , that defines the acceptable difference ratio between the diversity values predicted by model and the actual diversity values. We compare the

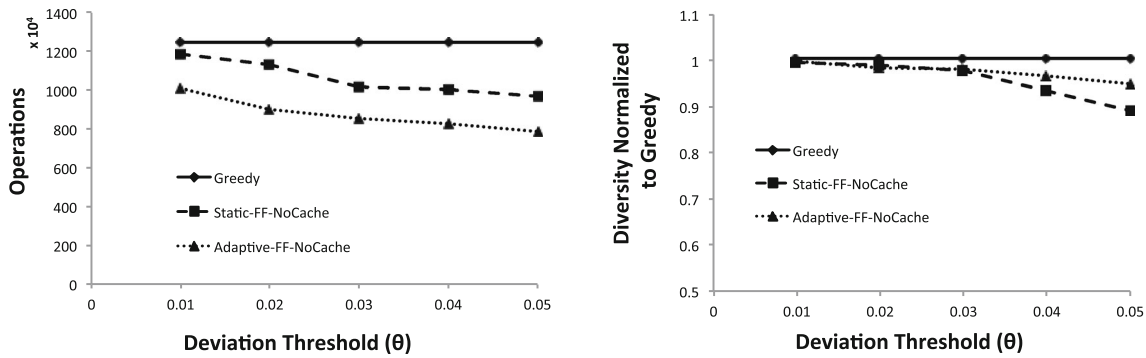


Fig. 10 Impact of θ on cost and diversity

performance of Adaptive-FF-no-cache scheme against Greedy heuristic and Static Model approach. Figure 9a shows that both Greedy and Static Model remains unaffected by the change in γ . However, as shown in Fig. 9a, as the threshold value is relaxed the adaptive model gets stable earlier and the predicted diversity values are used to locate diverse results earlier. This can be seen in higher cost savings as the value for γ increases. The savings in cost increase from 8 to 37% as the value of γ changes from 0.02 to 0.054. However, these savings in cost are at the expense of decrease in the diversity values of diverse subsets generated using adaptive model built with higher values of γ as shown in Fig. 9b. For $\gamma = 0.05$ the loss in diversity is upto 10%. The value of $\gamma = 0.03$ provides a good balance between cost savings and quality of diversification.

6.1.3 Impact of varying θ

In this experiment, the performance of Adaptive-FF-no-cache scheme is compared against Greedy heuristic and static model. The threshold value θ determines how close the diversity value of a subset should be from the predicted diversity to be acceptable. Figure 10a shows that as the threshold value is relaxed the cost savings for both the adaptive and static models as compared to the Greedy increase. For adaptive model these cost savings increase

from 20 to 40% and for the static model the increase is from 5 to 30%. The diversity values decrease upto 12% with increase in the threshold value θ as shown in Fig. 10b. This is due to the fact that results far from predicted diversity value are also included in the diverse subset. If the user prefer higher cost savings at the expense of slight decrease in quality of diversification, then higher values of θ are suitable; however, if quality of diversification is more important, that θ should be kept below 0.02.

6.2 Impact of using cached diverse results

In this experiment, we evaluate the impact of using cached diverse results from overlapping queries. In particular, we compare the performance of scheme that employs adaptive regression model without cached results against the schemes that use adaptive model with cached results. We have categorized this set of experiments into four sections by varying different experimental parameters.

6.2.1 Impact of Cached results in the absence of a Regression Model

In this experiment, we have focused on the impact of using cached diverse results in the absence of a regression model. We use a hypothetical setting that serves as a yard stick to measure the effectiveness of using cached diverse results in locating the diverse results for the current query. Thus, we assume the diverse subset for the current query is already evaluated using Greedy heuristic. Thus, the actual diversity of the subset S_c^i is known for all the values of $i, 2 \leq i \leq k$. These diversity values are used in each iteration i to locate the result that when added to S_c^{i-1} gives the same diversity value as S_c^i . We compare the performance of first-fit and best-fit diversification schemes. As shown in Fig. 11, the alternative schemes using cached diverse results are able to generate S_c in approximately 15% less number of operations as compared to FF-no-cache scheme that does not use any cached results. Also among different schemes FF-

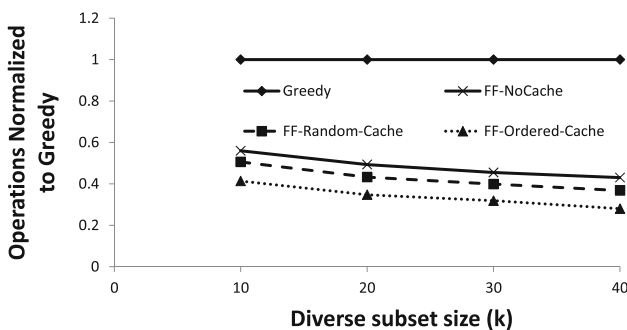


Fig. 11 Impact of cache without model

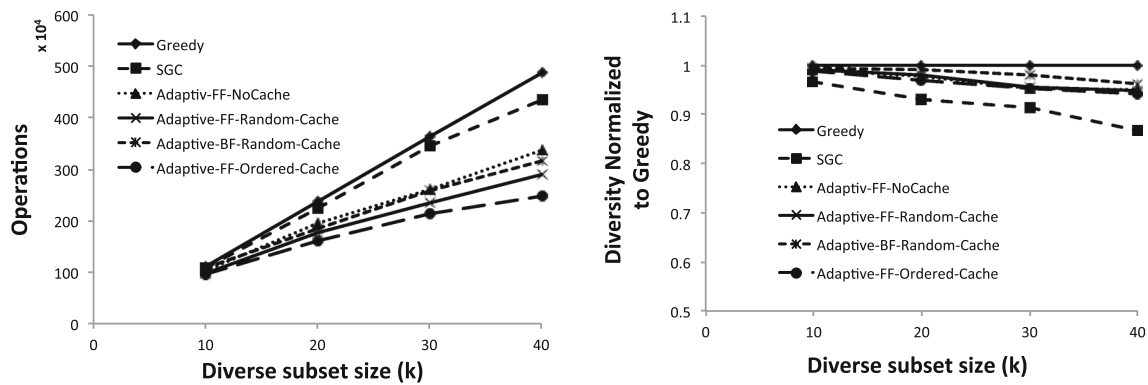


Fig. 12 Impact of varying k on cost and diversity

cache-ordered as discussed in Sect. 4.3.3 performs best in terms of cost. Further details on ordering cached results are given in Sect. 6.3

6.2.2 Impact of varying Diverse Set Size

In this experiment, we report on the impact of the required number of diverse results k . Figure 12a shows the number of operations (i.e. cost) performed by Greedy, SGC and AdOr schemes. As the figure shows, as the value of k increases, the cost increases for all schemes. AdOr, however, is performing up to 50% less operations than Greedy. Meanwhile, SGC reduces diversification cost by only 10% as compared to Greedy. This is due to the fact that SGC utilizes only limited number of cached diverse results that are obtained from only one previous overlapping query. Among the two variants of AdOr, Adaptive-FF (first-fit diversification) performs better in terms of cost as compared to Adaptive-BF (best-fit diversification). In particular, FF-Random-Cache reduces the cost by up to 10% compared to BF-Random-Cache. This is clearly because first-fit scheme terminates the search for an optimal result earlier without having to evaluate all the candidate results.

Further, Fig. 12a also shows that between the two variants of Adaptive-FF scheme, the Adaptive-FF-Ordered-Cache performs upto 9% less operations as compared to Adaptive-FF-Random-Cache. Thus, in terms of cost, Adaptive-FF-Ordered-Cache gives the best performance.

Figure 12b shows that the average diversity achieved by each scheme is decreasing with increasing the value of k . All AdOr schemes achieve comparable average diversity to Greedy. To highlight the benefits of AdOr in terms of achieved diversity, in Fig. 12b the diversity values are normalized to the diversity values achieved by Greedy heuristic. As the figure shows, the maximum loss in diversity for AdOr is within only 5% compared to Greedy whereas it goes upto 14% for SGC. As the value of k increases the loss in diversity for SGC increases as

it initializes diverse subset with higher number of overlapping results from the previous query without using any filtering.

Among the AdOr methods, Adaptive-BF performs better in terms of achieved diversity (Fig. 12b) because in each iteration, Adaptive-BF selects the one result providing the highest diversity if added to the diverse subset. Meanwhile, Adaptive-FF selects the first result that provides a diversity value within the threshold limit, thus introducing higher degree of approximation as compared to Adaptive-BF. However, as the regression model is used to control the degree of approximation, the loss in diversity between the two methods is within 2%.

6.2.3 Impact of Cache Size

To study the impact of cache size in terms of number of cached queries, we generate 100 random queries across various user sessions. Then, we vary the number of cached queries L from 20 to 100. Notice that in this experiment we are only evaluating the performance of AdOr schemes using cache, as SGC is not affected by the specified cache size. In terms of cost (i.e. number of operations), Adaptive-BF-Random-Cache exhibits an interesting pattern as shown in Fig. 13a. In that pattern, the cost of Adaptive-BF-Random-Cache decreases as the cache size increases up to a point, after which it starts increasing slightly. This is because at a moderate cache size, Adaptive-BF-Random-Cache has a higher chance to find a cached result that is close to optimal, while at the same time incurring a relatively low overhead in searching the cache. As the cache size increases, Adaptive-BF-Random-Cache still finds a cached result that is close to optimal, but it incurs a much higher cost in searching the rather large cache. Adaptive-FF-Random-Cache and Adaptive-FF-Ordered-Cache show a similar pattern, but are more resilient to the cached results as they terminate the search early. Adaptive-FF-Ordered-Cache

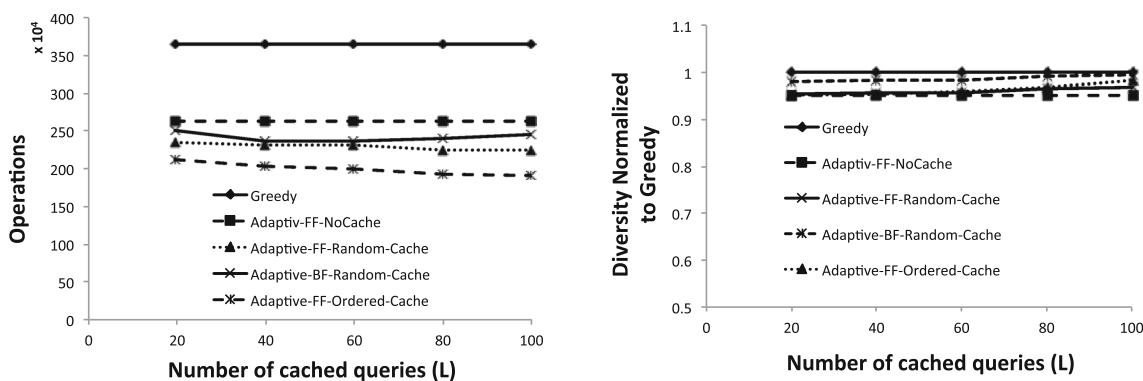


Fig. 13 Impact of cache size on cost and diversity

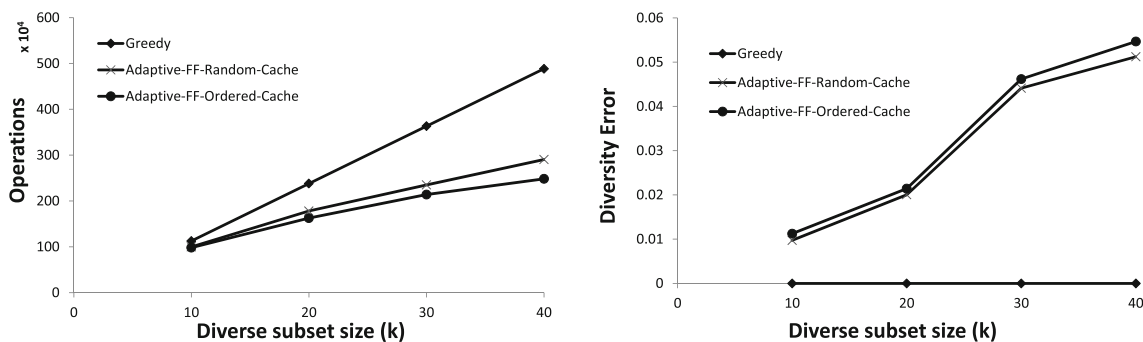


Fig. 14 Impact of ordering cache

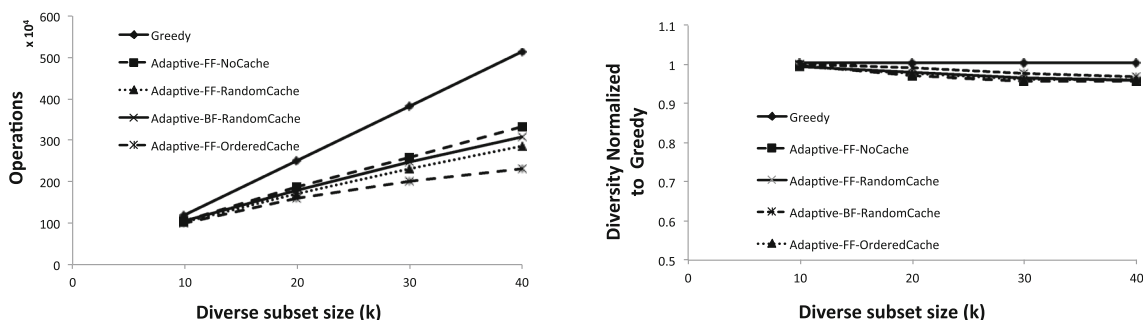


Fig. 15 Impact of varying k on cost and diversity (SDSS data set)

has additional overhead of computing set distances from centroids of overlapping queries in cache. Thus, as the cache size increases the difference in cost savings between Adaptive-FF-Random-Cache and Adaptive-FF-Ordered-Cache decreases to only 4%. Finally, Fig. 13b shows that AdOr consistently achieves diversity comparable to Greedy algorithm when varying the number of cached queries.

6.3 Impact of Ordering Cache

In this experiment we compare the performance of adaptive scheme that uses cached results without any priority and the scheme that reorders the cached results according to

their distances from the diverse subset of current query. Figure 14a shows that among different variations of adaptive scheme the Adaptive-FF-Ordered-Cache performs least number of operations. In particular, Adaptive-FF-Ordered-Cache performs 9% less operations as compared to Adaptive-FF-Random-Cache scheme. However, the diversity values of the subsets generated by both schemes are comparable as shown in Fig. 14b.

6.4 Results for SDSS Data Set

In this experiment, we report on the performance of AdOr scheme on the SDSS data set. Figure 15a, b shows that for the SDSS real data set, the performance of AdOr is

comparable to its performance on the uniform data set. For different values of diverse subset size k , AdOr outperforms Greedy algorithm in terms of number of operations with negligible loss in achieved diversity.

7 Related Work

In recent years, several diversification approaches have been proposed as in [21, 28–31]. Despite the considerable interest in diversification, most of the existing approaches consider diversification of a single query result. Recently, in [32] a new coverage-based definition of diversity is formulated for generating a combined diverse subset that recovers the results of multiple queries. However, our proposed AdOr scheme computes a separate diverse subset, using a traditional content-based definition of diversity that represents each query result set. Also, unlike [32], in our problem setting the queries are generated sequentially at different intervals of time. Therefore, the query results are not available simultaneously for generating a combined solution.

The central element of our approach is the use of a probabilistic model to estimate the diversity value of a diverse subset for future iterations of Greedy algorithm. Lots of work related to approximate query processing in database community make use of some form of probabilistic models (e.g. [24, 25]). However, to the best of our knowledge this is the first work where model-based approach is used in the post-query processing to diversify the query results.

Data caching and pre-fetching have also been shown as important approaches for reducing the cost of exploration queries [33]. In particular, caching has been used for efficient computation of representative data in interactive data exploration. For instance, in [34] a series of refined queries are evaluated by appropriately exploiting the information generated during the execution of previous queries in order to return top- K results. In [35], a caching mechanism is proposed that helps reduce the cost of computing future dynamic skyline queries by caching the results of previous skyline queries. While different caching approaches have been used in the literature for efficient representative data extraction, each approach varies in its methodology on “what to cache” and “how to use the cache” depending upon the end objective. Two closely related works using caching for search results diversification are presented in [21] and [28]. Specifically in [21], cached distance computations are used to reduce the cost of Greedy heuristic, whereas in [28], the idea of reusing already computed diverse results has been discussed. Especially, [28] extends the basic Greedy Construction heuristic for the case of continuous data streams. In particular, it perceives diversification as a continuous query, in which the k most diverse results need to be evaluated for each sliding window over

the data stream. Clearly, as the window slides over the data stream, some new data is added and some expire, leaving some significant overlap between any two consecutive windows. Similarly, in our problem setting each exploratory query can be perceived as a sliding window over the data space. Since different queries within an exploratory session typically explore the data space in a close vicinity to each other, it is very likely for two consecutive queries to have common results, similar to two consecutive sliding windows in a data stream. The premise underlying the proposed Stream Greedy Construction scheme in [28] is that instead of re-evaluating all the k diverse results for each sliding window, the diverse subset of the current window is initialized using the diverse results from the previous window. The drawback of this approach is the assumption that every window is uniformly populated from the data space. Thus, it is assumed that the valid diverse results from the previous window are still diverse with respect to the new data in the current window. However, in many data stream scenarios data distribution across different windows can be quite different. Thus, in our work we selectively choose only those diverse results from the cache that are still diverse with respect to the result of the current query. In Sect. 6, we have compared the performance of Stream Greedy Construction (SGC) against our proposed schemes.

8 Conclusions

Search result diversification has emerged as an important representative data extraction technique for interactive data exploration platforms. In order to reduce the overhead of diversification in an already computationally expensive exploration process, in this paper, we have presented the (*Adaptive model-based diversification AdOr*) scheme. AdOr targets the problem of efficiently diversifying the results of multiple queries within and across different exploratory sessions. Our novel scheme leverages the overlap between query results and utilizes an adaptive model-based approach that is particularly suitable for the efficient and effective diversification in IDE. AdOr provides solutions of quality comparable to existing baseline solutions, while significantly reducing processing costs. We present experimental results concerning the efficiency and the effectiveness of our approach on both synthetic and real data sets.

9 Appendix: Regression Model

In this paper we have used power function that can be represented as: $y = ax^{-b}$. a and b are the parameters we seek that would best fit the function to the sample data. These two parameters can be determined by using

nonlinear least squares analysis that is used to fit a set of m observations with a model that is nonlinear in n unknown parameters ($m > n$). The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations. Thus, the equation for squared error for N sample observations is presented as:

$$E^2 = \sum_{i=1}^N (y_i - f'(x_i))^2$$

The two parameters are found by partial derivative equations:

$$\frac{\partial E^2}{\partial a} = 0 \quad \text{and} \quad \frac{\partial E^2}{\partial b} = 0$$

which are solved simultaneously to obtain:

$$b = \frac{\sum x_i \ln(y_i) - \frac{1}{N} (\sum x_i) (\sum \ln(y_i))}{(\sum x_i^2) - \frac{1}{N} (\sum x_i)^2} \quad \text{and}$$

$$a = \exp \left[\frac{1}{N} \sum \ln(y_i) - b \frac{\sum x_i}{N} \right]$$

It has been shown that this yields a coefficient of determination of:

$$r^2 = \frac{[\sum x_i \ln(y_i) - \frac{1}{k} (\sum x_i) (\sum \ln(y_i))]^2}{\left[\sum x_i^2 - \frac{(\sum x_i)^2}{k} \right] \left[\sum \ln(y_i)^2 - \frac{(\sum \ln(y_i))^2}{k} \right]}$$

As in linear regression case, a value of $r^2 = 1$ infers a good fit of the model to the data.

Compliance with Ethical Standards

Conflict of interest None of the authors have any competing interests in this manuscript.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Çetintemel U et al. (2013) Query steering for interactive data exploration. In: CIDR
2. Kersten ML et al (2011) The researcher's guide to the data deluge: querying a scientific database in just a few seconds. PVLDB 4(12):1474–1477
3. Sellam T et al. (2013) Meet charles, big data query advisor. In: CIDR, USA
4. Banday AJ et al (2001) Mining the sky. MPA/ESO/MPE Workshop, New York
5. Ilyas IF et al (2008) A survey of top-k query processing techniques in relational database systems. ACM Comput Surv 40(4):11
6. Fagin R, Lotem A, Naor M (2001) Optimal aggregation algorithms for middleware. In: PODS
7. Tao Y et al (2007) Efficient skyline and top-k retrieval in subspaces. IEEE Trans Knowl Data Eng 19(8):1072–1088
8. Borzsony S et al. (2001) The skyline operator. In: ICDE
9. Acharya S et al. (1999) The aqua approximate query answering system. In: SIGMOD, USA, pp. 574–576
10. Agarwal S et al. (2014) Knowing when you're wrong: building fast and reliable approximate query processing systems. In: SIGMOD, USA, pp. 481–492
11. Berkhin P (2006) A survey of clustering data mining techniques. In: Grouping multidimensional data—recent advances in clustering, pp 25–71
12. Li C et al. (2007) Supporting ranking and clustering as generalized order-by and group-by. In: SIGMOD, China, pp 127–138
13. Singh M et al. (2016) DBExplorer: exploratory search in databases. In: EDBT, France, pp 89–100
14. Hina A Khan et al (2014) DivIDE: efficient diversification for interactive data exploration. In: SSDBM '14 Proceedings of the 26th International Conference on Scientific and Statistical Database Management, Aalborg, Denmark, 30 June–02 July, 2014, ACM, New York. doi:10.1145/2618243.2618253
15. Vieira MR et al. (2011) On query result diversification. In: ICDE
16. Clarke CLA et al. (2008) Novelty and diversity in information retrieval evaluation. In: SIGIR
17. Agrawal R et al. (2009) Diversifying search results. In: WSDM
18. Erkut E et al (1994) A comparison of p-dispersion heuristics. Comput Oper Res 21(10):1103–1113
19. Drosou M et al (2010) Search result diversification. SIGMOD Rec 39(1):41–47
20. Erkut E (1990) The discrete p-dispersion problem. Eur J Oper Res 46(1):48–60
21. Minack E et al. (2011) Incremental diversification for very large sets: a streaming-based approach. In: SIGIR
22. Chatzopoulou G et al (2011) The querie system for personalized query recommendations. IEEE Data Eng Bull 34(2):55–60
23. Dimitriadou K et al. (2014) Explore-by-example: an automatic query steering framework for interactive data exploration. In: SIGMOD, pp 517–528
24. Deshpande A et al. (2006) Mauvedb: supporting model-based user views in database systems. In: SIGMOD
25. Sharaf MA et al (2004) Balancing energy efficiency and quality of aggregate data in sensor networks. VLDB J 13(4):384–403
26. Boslaugh S (2012) Statistics in a nutshell, a desktop quick reference. 2nd edn. O'Reilly
27. Borodin A et al. (2012) Max-sum diversification, monotone submodular functions and dynamic updates. CoRR, vol. abs/1203.6397
28. Drosou M et al (2009) Diversity over continuous data. IEEE Data Eng Bull 32(4):49–56
29. Ziegler C-N et al. (2005) Improving recommendation lists through topic diversification. In: WWW
30. Vee E et al (2009) Efficient computation of diverse query results. IEEE Data Eng Bull 32(4):228
31. Khan HA et al. (2015) Progressive diversification for column-based data exploration platforms. In: ICDE, South Korea
32. Cheng S et al. (2014) Multi-query diversification in microblogging posts. In: EDBT
33. Idreos S et al. (2015) Overview of data exploration techniques. In: SIGMOD Australia
34. Chakrabarti K et al (2004) Evaluating refined queries in top-k retrieval systems. TKDE 16(2):256–270
35. Sacharidis D et al. (2008) Caching dynamic skyline queries. In: SSDBM, China