Efficient Evaluation of Object-Centric Exploration Queries for Visualization

You Wu* Duke University wuyou@cs.duke.edu
Boulos Harb Google Inc. harb@google.com
Jun Yang Duke University junyang@cs.duke.edu
Cong Yu Google Research congyu@google.com

ABSTRACT

The most effective way to explore data is through visualizing the results of exploration queries. For example, an exploration query could be an aggregate of some measures over time intervals, and a pattern or abnormality can be discovered through a time series plot of the query results. In this paper, we examine a special kind of exploration query, namely object-centric exploration query. Common examples include claims made about athletes in sports databases, such as "it is newsworthy that LeBron James has scored 35 or more points in nine consecutive games."

We focus on one common type of visualization, i.e., 2d scatter plot with heatmap. Namely, we consider exploration queries whose results can be plotted on a two-dimensional space, possibly with colors indicating object densities in regions. While we model results as pairs of numbers, the types of the queries are limited only by the users' imagination. In the LeBron James example above, the two dimensions are minimum points scored per game and number of consecutive games, respectively. It is easy to find other equally interesting dimensions, such as minimum rebounds per game or number of playoff games.

We formalize this problem and propose an efficient, interactive-speed algorithm that takes a user-provided exploration query (which can be a blackbox function) and produces an approximate visualization that preserves the two most important visual properties: the outliers and the overall distribution of all result points.

1 Introduction

Our work is partially motivated by observing claims made about interesting facts from data in the context of *computational journalism* [6, 5]. For example, *ESPN Elias Says...*[1] produces many factual claims based on players' performance statistics for a variety of professional sports in North America. While these claims come in very different forms, the key ingredient is comparison against claims in the same form. As an example, consider the following two claims about the performance of two NBA players.

- *Kevin Love's 31-point, 31-rebound game on Friday night ... Love became the first NBA player with a 30/30 game since Moses Malone had 38 points and 32 rebounds in a game back in 1982.*[2]
- *He (LeBron James) scored 35 or more points in nine consecutive games and joined Michael Jordan and Kobe Bryant as the only players since 1970 to accomplish the feat.*[3]

The purposes of both claims above are to highlight some player's performance, but they describe different aspects of the game. Common to both claims is the attempt to impress the reader by stating that few others have done the same or better before.

However, instead of just singling out outliers in text, it is more powerful to visualize the set of points representing all claims of the same form. Figure 1a shows one such visualization of all NBA players' `points` and `rebounds` stats in a single game by treating them as 2d points and plotting them in a scatter plot of the "sparse" points and a heatmap showing the density of the remaining points (we will define "sparse points" formally in Section 2.2). The visualization gives clear context on how impressive Kevin Love's 31/31 performance is by showing not only whether the performance is on the skyline, but also how far away it is from the edge of the cloud of mediocre performances. Furthermore, this single visualization can help the users explore many outstanding performances for which the same form of claims can be made, and see the distribution of players' performance in terms of `points-rebounds` in a single game. A similar visualization can be generated to evaluate LeBron James' 9-game 35-plus-point streak. Indeed, this visualization can be used for any form of claim that can be represented by a 2d scatter plot.

We identify two visual features essential to data exploration—outliers and clusters. This leads us naturally to the choice of overlaying a scatter plot (for outliers) on a heatmap (for clusters).

Given a large set $\mathbb{S}$ of points to visualize, we do not need to show the exact distribution of $\mathbb{S}$ because in practice because ordinary users are unable to perceive the difference between two dense regions containing a similar number of points (say 200 versus 210). Approximation can be easy in certain cases. For example, in the Kevin Love case, if the underlying data are stored as points and rebounds per player per game, each exploration query is a simple lookup. One can apply many existing outlier detection and density-estimation techniques to produce an approximation of the final plot. In many other cases, however, computing this visualization, even approximately, is a non-trivial task since it involves running many potentially expensive aggregate queries over the entire database. For example, in the LeBron James case, it takes an algorithm linear time to scan through a player's game-by-game scoring stats to

*Work partially done at Google Research New York.

[1] `http://espn.go.com/espn/elias`

Proceedings of the VLDB Endowment, Vol. 8, No. 12

[2] `http://espn.go.com/espn/elias?date=20101113`

[3] `http://www.nba.com/cavaliers/news/lbj_mvp_candidate_060419.html`

Notation	Description
$\mathcal{R}_i$	data for object i, as a sequence of tuples
n_i	number of tuples in $\mathcal{R}_i$
N	number of objects
$\mathcal{D}$	dataset, as a collection of data $\mathcal{R}_{1\ldots N}$ for objects $1 \ldots N$
$\|\mathcal{D}\|$	total number of tuples in $\mathcal{D}$
f	an exploration query
$\mathcal{S}_f(\mathcal{D})$	result of evaluating f on $\mathcal{D}$, as a multi-set of 2d points
$\mathcal{N}_{\mathbb{R}^2}(p;r)$	neighborhood of a 2d point p given radius r
$\mathcal{N}_{\mathcal{S}}(p;r)$	neighbors of a 2d point p in multi-set $\mathcal{S}$ given radius r
$\mathcal{S}_{\text{sparse}}$	points of $\mathcal{S}$ in a sparse neighborhood
$\mathcal{S}_{\text{sketch}}$	a "sketch" of $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$, as a set of weighted 2d points
r_x, r_y	neighborhood size
τ	neighborhood density threshold for points in $\mathcal{S}_{\text{sparse}}$

Table 1: Table of notations.

produce $\mathcal{S}$ representing his *prominent scoring streaks* [10]. In such cases, without knowing how the input is transformed into $\mathcal{S}$, it is impossible to sample directly from $\mathcal{S}$.

We observe that many datasets for exploration are comprised of data for objects (e.g., players or teams), and an exploration is often represented as a 2d point set $\mathcal{S}$ obtained by evaluating a blackbox exploration query on all the objects. It is clear that the objects do not contribute equally to the visual properties of the visualization of $\mathcal{S}$: many objects produce only points that will be buried inside dense regions, and are not interesting from the visualization perspective except that they contribute to the density color.

Consider a user-provided exploration query modeled as a function f that maps the data of each object to a set of 2d points, our goal is to, without knowing how f behaves, find a small set of objects, evaluate f on their data and produce a good approximation of the scatter plot with heatmap visualization of $\mathcal{S}$ and preserve the two aforementioned key visual features: outliers and clusters.

The main contributions of this paper include:

1. We formally define the two key visual features of scatter plot with heatmap type visualization, and quantify the quality of approximation.

2. We propose a two-phase sampling-based algorithm that efficiently generates an approximation of $\mathcal{S}$ for visualization without obtaining the exact $\mathcal{S}$ by evaluating f on the full data. Quantitative justification is provided.

3. We perform extensive experiments to evaluate both the quality of approximation and the efficiency of the sampling-based algorithm for interactive data exploration.

2 Exploration Query Problem

In this section, we introduce the basic notations (Section 2.1) and formally define the problem (Section 2.2). In Section 2.3, we present three examples of exploration queries that we study in this paper. For readers' convenience, we summarize all notations in Table 1.

2.1 Preliminaries

Suppose we have data $\mathcal{D}$ for N distinct objects as N relations $\mathcal{R}_1$, $\ldots, \mathcal{R}_N$. The data for object i, $\mathcal{R}_i = (t_{i,1}, \ldots, t_{i,n_i})$, is an ordered set of n_i tuples. All tuples $t_{i,j}$ conform to the same schema R. Let $\|\mathcal{D}\| = \sum_{i=1}^{N} n_i$ denote the total number of tuples for all objects.

Let f denote an *exploration query* that takes input an ordered set of tuples conforming to R, and outputs a (multi-)set of points in $\mathbb{R}^2$. The result of evaluating f on data $\mathcal{D} = \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}$ is denoted by $\mathcal{S}_f(\mathcal{D})$, or simply $\mathcal{S}$ when the context is clear. $\mathcal{S}_f(\mathcal{D})$ is defined as the bag of the results of evaluating f on each $\mathcal{R}_i$, i.e., $\mathcal{S}_f(\mathcal{D}) = \uplus_{i=1}^{N} f(\mathcal{R}_i)$, or simply $\mathcal{S}$ when the context is clear. We formally define the *neighborhood* of a point $p \in \mathbb{R}^2$ as

$$\mathcal{N}_{\mathbb{R}^2}(p;r) = \{p' \in \mathbb{R}^2 \mid \|p - p'\| \le r\},$$

with the radius r as an input.

The *neighbors* of a point p in a finite set $\mathcal{S}$ is denoted by

$$\mathcal{N}_{\mathcal{S}}(p;r) = \mathcal{N}_{\mathbb{R}^2}(p;r) \cap \mathcal{S}.$$

In the remainder of the paper, we use (scaled) L_∞-norm for $\|\cdot\|$ in the neighbor definition above, i.e.:

$$\mathcal{N}_{\mathbb{R}^2}(p;r_x,r_y) = \{p' \in \mathbb{R}^2 \mid |p.x - p'.x| \le r_x \wedge |p.y - p'.y| \le r_y\};$$
$$\mathcal{N}_{\mathcal{S}}(p;r_x,r_y) = \mathcal{N}_{\mathbb{R}^2}(p;r_x,r_y) \cap \mathcal{S}.$$

However, the results of this paper extend to other commonly used norms as well, e.g., L_1-norm, L_2-norm.

2.2 Problem Definition

Given an exploration query f on dataset $\mathcal{D}$, our goal is to efficiently generate the following two components for visualizing $\mathcal{S}_f(\mathcal{D})$.

- $\mathcal{S}_{\text{sparse}}$: points of $\mathcal{S}_f(\mathcal{D})$ in a sparse neighborhood (for scatter plot);
- $\mathcal{S}_{\text{sketch}}$: a "sketch" of rest of the points, as a set of weighted points (for heatmap).

We define the *sparse points* $\mathcal{S}_{\text{sparse}}$ as the set of points whose number of neighbors is no more than some *sparsity threshold* τ. Given r_x and r_y that define the neighborhood of a point and sparsity threshold τ, all points with a sparse neighborhood can be precisely identified.

For all the other points, i.e., $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$, we create a *sketch* $\mathcal{S}_{\text{sketch}}$ (not to be confused with the concept of sketch in data streams). The sketch consists of a set of weighted points, where $(p, w_p) \in \mathcal{S}_{\text{sketch}}$ represents an estimated w_p points of $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$ in p's neighborhood. A good sketch should have each point of $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$ "covered" by one point of $\mathcal{S}_{\text{sketch}}$. We use the *sketch distance function* δ, which will be defined later in this section, to capture the quality of a sketch $\mathcal{S}_{\text{sketch}}$ measured w.r.t. $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$.

The challenge in evaluating an exploration query comes with a computation budget. To serve the purpose of realtime exploration, we would like to evaluate the query f without performing the evaluation on the full data of all objects. With a limited amount of data accesses, an approximate solution $(\tilde{\mathcal{S}}_{\text{sparse}}, \mathcal{S}_{\text{sketch}})$ needs to be found. The quality of $\tilde{\mathcal{S}}_{\text{sparse}}$ is measured by its precision and recall w.r.t. $\mathcal{S}_{\text{sparse}}$, while the quality of $\mathcal{S}_{\text{sketch}}$ is measured by δ w.r.t. $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$.

Motivated by achieving as high quality as possible for both $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ within a budget constraint, we formally define the exploration query evaluation problem as follows.

Definition 1 (Exploration Query Evaluation)**.** *With a relational schema R, a dataset of N objects $\mathcal{D} = \{\mathcal{R}_i\}_{i=1}^{N}$, the full result of an exploration query f is given by $\mathcal{S} = \uplus_{i=1}^{N} f(\mathcal{R}_i)$.*

Given neighborhood radius r_x, r_y, a sparsity threshold τ, solve the following two tasks:

- ***Task 1.*** *Find $\tilde{\mathcal{S}}_{sparse}$ that approximates the set of sparse point $\mathcal{S}_{sparse} \subseteq \mathcal{S}$, where*

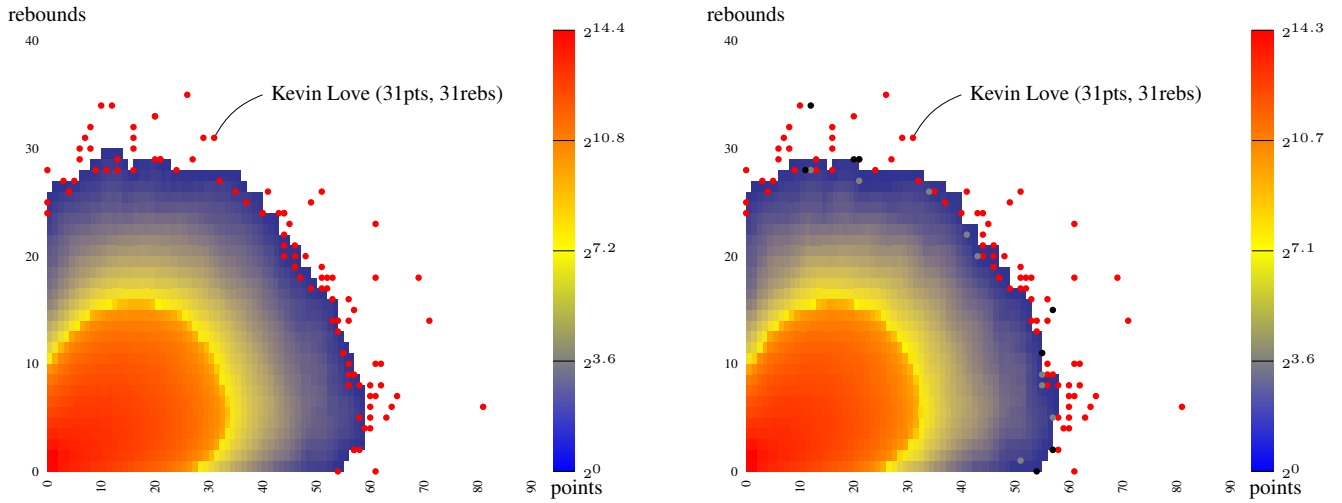
$$\mathcal{S}_{sparse} = \{p \in \mathcal{S} \mid |\mathcal{N}_{\mathcal{S}}(p;r_x,r_y)| \le \tau\}.$$

- ***Task 2.*** *Find a weighted point set*

$$\mathcal{S}_{sketch} = \{(p, w_p) \mid p \in \mathcal{S} \wedge w_p \in \mathbb{Z}^+\}$$

that minimizes $\delta(\mathcal{S} \setminus \mathcal{S}_{sparse}, \mathcal{S}_{sketch})$.

subject to a given computation budget $\eta \in (0, 1]$ such that at most $\eta \cdot \|\mathcal{D}\|$ tuples can be accessed during evaluation.



(a) $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ from full result.

(b) $\tilde{\mathcal{S}}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ from an approximate solution: ● true positive; ● false positive; ● false negative (all w.r.t. $\mathcal{S}_{\text{sparse}}$).

Figure 1: Projection query example on (`points`, `rebounds`), visualized using 2d scatter plot for $\mathcal{S}_{\text{sparse}}$ and $\tilde{\mathcal{S}}_{\text{sparse}}$, on top of heatmap for $\mathcal{S}_{\text{sketch}}$. In the heatmap, the weight of each point is distributed uniformly in its neighborhood.

Sketch Distance. The distance function δ measures the quality of the sketch $\mathcal{S}_{\text{sketch}}$ w.r.t. $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$ and needs to capture two aspects of sketch quality:

I. the *distribution* of $\mathcal{S}_{\text{sketch}}$ should resemble that of $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$;

II. the *magnitude* of $\mathcal{S}_{\text{sketch}}$ should be close to that of $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$.

Definition 2 (Sketch Distance)**.** *Given a multiset of points* $P = \{p_1, p_2, \ldots, p_n\}$ *and a weighted point set* $Q = \{(q_1, w_1), (q_2, w_2), \ldots, (q_m, w_m)\}$*, the* sketch distance[4] *between* P *and* Q *is defined as*

$$\delta(P, Q) = 1 - \frac{\mathsf{OPT}}{\max\{|P|, \|Q\|\}},$$

where $\|Q\| = \sum_{j=1}^{m} w_j$ *and* OPT *is the optimal solution to the following integer program:*

$$\textit{maximize} \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}\mathbf{1}[p_j \in \mathcal{N}_{\mathbb{R}^2}(q_i; r_x, r_y)], \tag{1}$$

$$\textit{subject to} \sum_{j=1}^{n} x_{ij} \le w_i,\ 1 \le i \le m \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} \le 1,\ 1 \le i \le n \tag{3}$$

$$x_{ij} \in \{0, 1\},\ 1 \le j \le n,\ 1 \le i \le m. \tag{4}$$

Consider P as the set of points to be sketched, i.e., $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$, and the weighted point set Q as the sketch. A point $(q, w) \in Q$ can cover a point $p \in P$ if p and q are neighbors, and q can cover at most w points in P. On the other hand, each point of p can be covered by at most one point of Q. Hence, OPT is the maximum number of points of P that Q can cover.

The quantity $\frac{\mathsf{OPT}}{\max\{|P|, \|Q\|\}}$ measures the similarity between P and sketch Q. Dividing by the larger of $|P|$ and $\|Q\|$ penalizes possible disparity between them.

Let us revisit Kevin Love's 31-point, 31-rebound game example. Consider the performance of all NBA players in all games in terms of point-rebound, and visualize all points ($\approx 10^6$ of them) by combining $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$.[5] In Figure 1a, points of $\mathcal{S}_{\text{sparse}}$ are plotted as red dots. $\mathcal{S}_{\text{sketch}}$ is visualized using heatmap, by distributing the weight of each point evenly in its neighborhood. Here, $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ are produced using an expensive baseline algorithm (Algorithm 1), which performs full evaluation on the entire dataset $\mathcal{D}$ and guarantees $\delta(\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}, \mathcal{S}_{\text{sketch}}) = 0$.

On the other hand, Figure 1b visualizes, in the same way, approximated sets $\tilde{\mathcal{S}}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ produced by Algorithm 2, which accesses only 20% tuples of the entire dataset, with the projection query passed in as a blackbox function. To compare with the visualization on the full result, we show false positives and false negatives of $\tilde{\mathcal{S}}_{\text{sparse}}$ w.r.t. $\mathcal{S}_{\text{sparse}}$ as grey and black dots, respectively. It is obvious in both Figure 1a and 1b that Love's 31-point, 31-rebound performance was impressive, being far from the vast majority of all points. One can also see the resemblance between the two figures, even though Figure 1b incurs significantly fewer data accesses and hence much lower latency.

2.3 Query Types

We study the following three types of exploration queries commonly made on sports data. In Section 6.1, we also show applications to two other domains, the *Computer Science bibliography* and the *Wikipedia edit history*.

I. **Projection Query.** Given numerical attributes $A, B \in R$,

$$f(\mathcal{R}) = \{(t.A, t.B) \mid t \in \mathcal{R}\}.$$

An example application of the projection query would be to find a player's point-rebound stats in every game; i.e., $(A, B) = (\texttt{points}, \texttt{rebounds})$. When f is applied to Kevin Love, one of the result 2d points would be $(31, 31)$, which corresponds to "Kevin Love's 31-point, 31-rebound game."

II. **Count Query.** Given numerical attribute $A \in R$,

$$f(\mathcal{R}) = \{(v, c) \mid v \in \mathcal{R}.A \wedge c = |\{t \in \mathcal{R} \mid t.A \ge v\}|\}.$$

[4]The sketch distance is adapted from the Earth Mover's Distance [15] widely used in measuring the dissimilarity between two images in image processing. Adaptations are made to suit the purpose of this work.

[5]We set $r_x = r_y = 2$ and $\tau = 16$ such that $\mathcal{S}_{\text{sparse}}$ contains roughly 100 points so that human eye can perceive.

For example, Michael Jordan's 38 games with 50 or more points (most in NBA history[6]) maps to a 2d point $(50, 38)$ when evaluating a count query on Jordan's game-by-game stats data with $A = \texttt{points}$. Note that f can return multiple points for Michael Jordan, with lower point values associated with higher count values.

III. **Streak Query.** Given numerical attribute $A \in R$,

$$f(\mathcal{R}) = \text{Pareto optimal subset of } \{(v, \mathsf{len} = r - l + 1) \mid 1 \le l \le r \le n_{\mathcal{R}} \ \wedge \ v = \min_{l \le i \le r} t_i.A\}.$$

This is known as the set of *prominent streaks* in sequence $\mathcal{R}.A$ [10]. For instance, LeBron James's 9-game 35-or-more-points streak is represented by a 2d point $(35, 9)$ in the result set of evaluating a streak query on James' stats (with $A = \texttt{points}$).

While these three types of exploration queries are represented by very different functions f, they share one common characteristic—same formats of input (a relation conforming to schema R) and output (a set of points in $\mathbb{R}^2$). In the rest of the paper, we illustrate our algorithms using these query types.

3 System Overview

In this section, we first describe our storage structure for data, and then walk through the flow of the algorithm.

3.1 Data Storage and Working Memory

From a system point of view, we would like to host the exploration query evaluation service for multiple datasets. Hence, we will not dedicate memory to storing the entire dataset, even if it fits into memory by itself. Instead, the dataset is stored in SSTable [4], a distributed key-value storage system supporting efficient lookup. The data of an object is accessed via a service API using its index as the key; additionally, *any single tuple* can be accessed using the object index along with the tuple index as the key. In other words, the API allows both object- and tuple-level access.

On the other hand, we do assume that a small amount of memory is reserved for storing a small sample of the dataset, and that enough working memory is available, on a per-query basis, for storing data accessed during query evaluation (whose size is capped by budget η) and for query evaluation.

3.2 Workflow

Here we describe the high-level workflow of evaluating an exploration query (Figure 2). Detailed algorithms and analysis will be provided in Sections 4 and 5, respectively.

0. Upon initialization of the query evaluator, we establish the connection to the dataset $\mathcal{D}$, and "prefetch" a random sample of $\zeta\|\mathcal{D}\|$ tuples into memory, where ζ is the sample rate and $\|\mathcal{D}\|$ is the total number of tuples in $\mathcal{D}$. We leave the the details of this prefetching step to Section 4.

   Since the prefetching step is performed only once and its cost is amortized over all ensuing exploration queries regardless of their types, we do not count its data accesses towards the budget η in our problem definition. On the other hand, the sample rate ζ is bounded by the amount of memory reserved for storing the sample, which is far less than the total data size.

[6] `www.nba.com/jordan/list_50games.html`

Algorithm 1: ExploreBase(f, r_x, r_y, τ)

```
1  S ← ∅;
2  for i = 1 to N do
3  |   R_i ← LoadObjectData(i);
4  |   S ← S ⊎ f(R_i);
5  S_sparse ← ∅; S_sketch ← ∅;
6  foreach p ∈ S do
7  |   if |N_S(p; r_x, r_y)| ≤ τ then
8  |   |   S_sparse ← S_sparse ⊎ {p};
9  |   else
10 |   |   S_sketch ← S_sketch ⊎ {(p, 1)};
11 return S_sparse, S_sketch;
```

1. When an exploration query comes in, f is passed in as a blackbox function, with attribute(s) specified as parameter(s) of f.

2. At its discretion, the algorithm executes function f on the prefetched sample, and/or full data for a subset of the objects retrieved via the object-level API. Depending on the algorithm, this step may be performed more than once.

3. Based on the results of executing f in the previous step, the algorithm computes the set $\mathcal{S}_{\text{sparse}}$ of sparse points and a sketch of the remaining points $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$, or approximations thereof.

4. The results are then combined to produce a visualization for the result of the exploration query, using a scatter plot overlaid on top of a heatmap.

In this workflow, the evaluator assumes that it knows only the input and output format of f, and nothing about how f actually processes the input and produces the output. In other words, an exploration query can be represented by any blackbox function f carrying the signature specified in Section 2.1, with a few examples given in Section 2.3.

4 Algorithms

In this section, we present two algorithms. The baseline algorithm (Algorithm 1) performs exact evaluation of an exploration query f, ignoring computation budget η, and computes accurate ($\mathcal{S}_{\text{sparse}}$, $\mathcal{S}_{\text{sketch}}$) for the full result set $\mathcal{S}$. The sampling-based algorithm (Algorithm 2) produces approximate solutions to $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ more efficiently than the baseline, within the computation budget η.

4.1 Baseline Algorithm

We first present a straightforward algorithm (Algorithm 1) that performs exact evaluation on the full dataset given an exploration query f. The algorithm takes as input (i) a callable function f specifying the exploration query, and (ii) radii r_x and r_y that define the neighborhood of a point, and sparsity threshold τ.

Algorithm 1 evaluates the exploration query in a brute-force fashion. For each object i, we load its data $\mathcal{R}_i$ into memory (line 3). (If a tuple is in the prefetched sample, we can avoid reloading it; this detail is not shown in Algorithm 1.) Then, we feed object i's data as input to function f to evaluate (line 4). $\mathcal{S}$ is simply the union of execution output over all objects. For each point $p \in \mathcal{S}$, we include it in $\mathcal{S}_{\text{sparse}}$ if it has no more than τ neighbors in $\mathcal{S}$. Otherwise, we simply create a sketch point for it with weight 1 in $\mathcal{S}_{\text{sketch}}$.

Counting Neighbors. Neighbor counting by brute force can take as much as $O(|\mathcal{S}|^2)$ time. We use the following heuristic by partitioning the result set $\mathcal{S}$ using grid cells each of size $r_x \times r_y$.

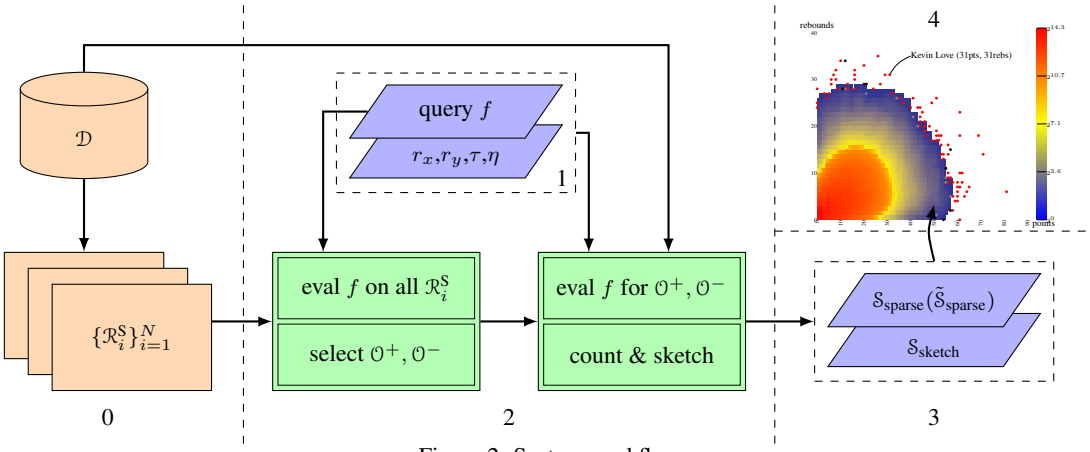


Figure 2: System workflow.

By Lemma 1, the neighborhood of a point $p \in \mathbb{R}^2$ contains the grid cell $\square_{ij}$ containing p, and is contained by the 3×3 grid cells centered at $\square_{ij}$. Moving from $\mathbb{R}^2$ to the finite result set $\mathcal{S}$, for a $p \in \mathcal{S}$ that belongs to some partition $\mathcal{P}_{ij} = \mathcal{S} \cap \square_{ij}$, p's neighbors in $\mathcal{S}$, i.e., $\mathcal{N}_{\mathcal{S}}(p; r_x, r_y)$, form a superset of $\mathcal{P}_{ij}$ and fall within the 3×3 partitions centered at $\mathcal{P}_{ij}$ (Corollary 1).

Lemma 1. *For any $p \in \mathbb{R}^2$, if $p \in \square_{ij}$, where $\square_{ij} = [i \cdot r_x, (i+1) \cdot r_x) \times [j \cdot r_y, (j+1) \cdot r_y)$, then we have*

i. $\square_{ij} \subseteq \mathcal{N}_{\mathbb{R}^2}(p; r_x, r_y)$;

ii. $\mathcal{N}_{\mathbb{R}^2}(p; r_x, r_y) \subseteq \bigcup_{i'=i-1}^{i+1} \bigcup_{j'=j-1}^{j+1} \square_{i'j'}$.

Corollary 1. *Given $\mathcal{S} \subseteq \mathbb{R}^2$, partition it as $\mathcal{S} = \bigcup_{i,j \in \mathbb{Z}} \mathcal{P}_{ij}$, where $\mathcal{P}_{ij} = \mathcal{S} \cap \square_{ij}$. For any $p \in \mathcal{P}_{ij}$, we have*

i. $\mathcal{P}_{ij} \subseteq \mathcal{N}_{\mathcal{S}}(p; r_x, r_y)$;

ii. $\mathcal{N}_{\mathcal{S}}(p; r_x, r_y) \subseteq \bigcup_{i'=i-1}^{i+1} \bigcup_{j'=j-1}^{j+1} \mathcal{P}_{i'j'}$.

Because $\mathcal{S}$ is determined by f, whose behavior is assumed to be unknown to the algorithm, it is not possible to index $\mathcal{S}$ beforehand in order to speed up neighbor counting. However, with Corollary 1, we can avoid performing $O(|\mathcal{S}|^2)$ comparisons, and narrow down the possible neighbors of a point to its 9 adjacent partitions. Moreover, if a partition $\mathcal{P}_{ij}$ contains more than τ points, we can immediately determine that all its points should be added to the sketch.

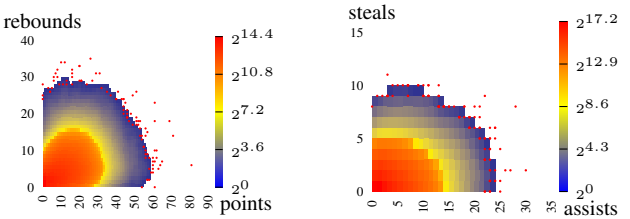
Time Complexity. The execution of Algorithm 1 consists of three steps: (i) loading each object's data into memory, (ii) executing f on each object's data, and (iii) computing $\mathcal{S}_{\text{sparse}}$ and $\mathcal{S}_{\text{sketch}}$ from $\mathcal{S}$.

Step (i) requires accessing the full dataset, which requires fetching $(1-\zeta)\|\mathcal{D}\|$ tuples, with prefetched ones excluded. This use of the prefetched sample is not very effective; we will see a much better use of this sample in Section 4.2.

Steps (ii) and (iii) are carried out in memory. Step (ii) depends on how f behaves, while step (iii) only depends on the size of the result set $\mathcal{S}$ and the sparsity threshold τ.

For step (ii), the time complexity for the brute-force execution of f is linear in the number of tuples for all three types of exploration queries described in Section 2.3. Hence, the overall complexity of executing such queries on $\mathcal{D}$ is $O(\|\mathcal{D}\|)$.

For step (iii), thanks to the counting technique we have described, the worse case complexity is improved from $O(|\mathcal{S}|^2)$ to $O(\tau \cdot |\mathcal{S}|)$. The worse-case scenario is that each (non-empty) partition of $\mathcal{S}$ contains τ points (so no point can be pruned for counting), and each non-empty partition is adjacent to one or more (at most 8) other non-empty partition(s). In this case, the total number of pairs



(a) f_1 : Projection on `points-rebounds` plain

(b) f_2 : Projection on `assists-steals` plain

Figure 3: Comparing results of projection queries with different attributes.

of points compared is $O\left(\tau^2 \cdot \frac{|\mathcal{S}|}{\tau}\right) = O(\tau \cdot |\mathcal{S}|)$. In particular, for all three types of exploration queries we consider, $|\mathcal{S}| = O(\|\mathcal{D}\|)$, i.e., linear in the size of the full dataset $\mathcal{D}$.

Result Quality. In terms of the quality of the output, without imposing the computation budget η, Algorithm 1 trivially returns the exact $\mathcal{S}_{\text{sparse}}$ and a sketch that gives $\delta(\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}, \mathcal{S}_{\text{sketch}}) = 0$.

4.2 Sampling-based Algorithms

The baseline algorithm essentially fetches the full dataset $\mathcal{D}$, object by object. Due to the large volume of the data, accessing the entire dataset piece by piece via service API can be very costly. The complexity of full evaluation and counting makes it inefficient to perform such brute-force evaluation in an interactive environment.

The baseline algorithm also disregards the computation budget η. Working under a budget constraint is challenging. Because we do not assume anything about the behavior of f, there is no guarantee that evaluating f on partial data of an object will produce a partial output of evaluating f on the object's full data. Therefore, in order to comply with the budget constraint, we must choose to evaluate f on the full data for a subset of objects, and completely ignore the remaining objects. But how can we know which subset of objects to choose, without knowing how f behaves? The challenge is compounded by the problem that the data $\mathcal{R}_i$ of each object i follows some unknown distribution. For example, in the context of the basketball data, an object could be a player or a team. For a player object, depending on the players' position on the court (e.g., point guard vs. power forward), and depending on the ability of the player (e.g., a superstar like Michael Jordan vs. a mediocre player), the distribution of this object's data will obviously be very different from other objects.

Fortunately, the prefetched sample comes to rescue. Suppose that for each object, we have a sample of its data. Then, the result

Algorithm 2: ExploreSample($f, \{\mathcal{R}_i^{\mathbb{S}}\}_{i=1}^N, r_x, r_y, \tau, \eta$)

1 $\mathbb{S}^{\mathbb{S}} \leftarrow \varnothing$;
2 **for** $i = 1$ **to** N **do**
3 $\quad \mathbb{S}^{\mathbb{S}} \leftarrow \mathbb{S}^{\mathbb{S}} \uplus f(\mathcal{R}_i^{\mathbb{S}})$;
4 $(\mathcal{O}^+, \mathcal{O}^-) \leftarrow$ SelectObjects($\mathbb{S}^{\mathbb{S}}, \eta$);
5 $\mathbb{S}^+ \leftarrow \varnothing$;
6 **foreach** $i \in \mathcal{O}^+$ **do**
7 $\quad \mathcal{R}_i \leftarrow$ LoadObjectData(i);
8 $\quad \mathbb{S}^+ \leftarrow \mathbb{S}^+ \uplus f(\mathcal{R}_i)$;
9 $\mathbb{S}^- \leftarrow \varnothing$;
10 **foreach** $j \in \mathcal{O}^-$ **do**
11 $\quad \mathcal{R}_j \leftarrow$ LoadObjectData(j);
12 $\quad \mathbb{S}^- \leftarrow \mathbb{S}^- \uplus f(\mathcal{R}_j)$;
13 $\tilde{\mathbb{S}}_{\text{sparse}} \leftarrow \varnothing$; $\mathbb{S}_{\text{sketch}} \leftarrow \varnothing$;
14 **foreach** $p \in \mathbb{S}^+$ **do**
15 $\quad$ **if** $|\mathcal{N}_{\mathbb{S}^+}(p; r_x, r_y)| + \lambda \cdot |\mathcal{N}_{\mathbb{S}^-}(p; r_x, r_y)| \leq \tau$ **then**
16 $\quad\quad \tilde{\mathbb{S}}_{\text{sparse}} \leftarrow \tilde{\mathbb{S}}_{\text{sparse}} \uplus \{p\}$;
17 $\quad$ **else**
18 $\quad\quad \mathbb{S}_{\text{sketch}} \leftarrow \mathbb{S}_{\text{sketch}} \uplus \{p, 1\}$;
19 **foreach** $p \in \mathbb{S}^-$ **do**
20 $\quad \mathbb{S}_{\text{sketch}} \leftarrow \mathbb{S}_{\text{sketch}} \uplus \{p, \lambda\}$;
21 **return** $\tilde{\mathbb{S}}_{\text{sparse}}, \mathbb{S}_{\text{sketch}}$;

of applying f on the prefetched sample may resemble the full result $\mathbb{S}$ in terms of outlier identities, even though the outliers returned by f, e.g., a *streak query*, represent low-probability events that come from one or a few tuples of the full data. In Section 5.1, we provide an analysis of this connection between the result on the sample and that on the full data for *projection queries*.

In the remainder of this section, we describe a general sampling-based algorithm that uses the prefetched sample to select a small number of objects, for which we access their full data for evaluation (Algorithm 2).

Prefetching. Algorithm 2 assumes a prefetching step (Step 0 described in Section 3.2) that works as follows. Upon initialization of the query evaluator, given a fixed sample rate ζ, for each object i, we sample ζn_i times uniformly and independently at random from $t_1, \ldots, t_{n_i}$ with replacement. All the ζn_i sample tuples form the sample data $\mathcal{R}_i^{\mathbb{S}}$ of object i. The full set of sample data $\{\mathcal{R}_i^{\mathbb{S}}\}_{i=1}^N$ sits in the memory throughout the lifetime of the evaluator, and is fed into the evaluation of each forthcoming exploration query f in Algorithm 2.

Objects Selection. Algorithm 2 first executes f on the prefetched sample for all objects. The result, denoted by $\mathbb{S}^{\mathbb{S}}$, is the union of $f(\mathbb{S}_i^{\mathbb{S}})$ along with the ownership of each point in the result (line 3). Base on $\mathbb{S}^{\mathbb{S}}$, we select two disjoint subsets of objects (line 4): (i) $\mathcal{O}^+$, the set of objects such that we envision for each $p \in \mathbb{S}_{\text{sparse}}$, the object i that yields p under f (i.e., $p \in f(\mathcal{R}_i)$) belongs to $\mathcal{O}^+$, and (ii) $\mathcal{O}^-$, a random sample of all objects $\mathcal{O}$ excluding $\mathcal{O}^+$ (i.e., $\mathcal{O} \setminus \mathcal{O}^+$). We ensure that the total number of tuples contained in the full data for these objects is within the budget η. We defer the discussion of how the objects are selected (line 4) to Section 5.

Full Execution. The two sets of objects $\mathcal{O}^+$ and $\mathcal{O}^-$ are presumably much smaller than the set of all objects. Algorithm 2 then executes f on the full data for $\mathcal{O}^+$ and $\mathcal{O}^-$; we denote the result sets as $\mathbb{S}^+$ and $\mathbb{S}^-$ respectively.

Counting. There are two key differences in neighbor counting and result generation compared to Algorithm 1: (1) only points of $\mathbb{S}^+$ may be included in $\tilde{\mathbb{S}}_{\text{sparse}}$, and (2) while each point of $\mathbb{S}^+$ is counted once as before, the presence of each point in $\mathbb{S}^-$ is multiplied by λ, a multiplier depending on how the budget η is divided between $\mathcal{O}^+$ and $\mathcal{O}^-$. The choice of the value of λ will also be described in Section 5.

Time Complexity. Compared to the complexity of the Algorithm 1, Algorithm 2 reduces the cost of fetching data and counting from $O(\|\mathcal{D}\|)$ to $O((\zeta+\eta)\cdot\|\mathcal{D}\|)$. For cost of execution f for a query with linear/super-linear complexity $T(n)$, the overall time complexity is reduced to $T((\zeta + \eta) \cdot \|\mathcal{D}\|)$. Specifically, for the three types of queries we consider with linear time complexity, the overall time complexity is also reduced to $O((\zeta + \eta) \cdot \|\mathcal{D}\|)$.

Difficulty of Provisioning $\mathcal{O}^+$. It is obvious that the choice of $\mathcal{O}^+$ is critical to the quality of $\tilde{\mathbb{S}}_{\text{sparse}}$—if $i \notin \mathcal{O}^+$, there is no chance for points of $f(\mathcal{R}_i)$ to appear in $\tilde{\mathbb{S}}_{\text{sparse}}$. To understand why it is necessary to perform online selection of $\mathcal{O}^+$ and $\mathcal{O}^-$ based on $\mathbb{S}^{\mathbb{S}}$, one must consider the diversity of queries that can be applied to a single dataset.

Figure 3 shows the scatter plot with heatmap of accurate results for two queries of the same type (projection) but on different attributes. Query f_1 projects the NBA players' game-by-game performance to the `points-rebounds` plane, while f_2 projects the same data on two different attributes `assists` and `steals`. For each of f_1 and f_2, we use a rectangular neighborhood of size approximately $\frac{1}{10} \times \frac{1}{10}$ of the result space, and a proper value of τ that limits the size of $\mathbb{S}_{\text{sparse}}$ to be roughly 100. Comparing $\mathbb{S}^1_{\text{sparse}}$ for f_1 and $\mathbb{S}^2_{\text{sparse}}$ for f_2, $\mathbb{S}^1_{\text{sparse}}$ has 99 points (possibly overlapping) corresponding to 40 distinct objects, and $\mathbb{S}^2_{\text{sparse}}$ has 101 points corresponding to 46 distinct objects. Together, $\mathbb{S}^1_{\text{sparse}}$ and $\mathbb{S}^2_{\text{sparse}}$ consist of a total of 83 distinct objects, sharing only 3 in common.

This example illustrates that it is impossible to use a static choice of $\mathcal{O}^+$ to provide a good coverage of objects that result in points of $\mathbb{S}_{\text{sparse}}$ for all possible queries. Therefore, we perform query-specific online object selection, as we explain in the next section.

5 Objects Selection

In this section, we discuss how $\mathcal{O}^+$ and $\mathcal{O}^-$ are selected in Algorithm 2 (line 4).

At a high level, this process is analogous to the problem of computing a small *coreset* of a large point set that can be used for approximating various "extent measures" of the full point set [1]. More specifically, for the exploration query evaluation problem, we would like to approximate two types of extent measures—the sparse points $\mathbb{S}_{\text{sparse}} \subseteq \mathbb{S}$, and a density estimate for all other points $\mathbb{S} \setminus \mathbb{S}_{\text{sparse}}$. The challenge here is that we do not have direct access to the full point set $\mathbb{S}$. For example, as we treat the evaluation function f as a blackbox, there is no way to sample points uniformly and independently from $\mathbb{S}$ without computing it first, i.e., performing full evaluation on all objects. In other words, with an f whose behavior is unknown to Algorithm 2, the coreset we construct cannot be an arbitrary subset of $\mathbb{S}$. It has to be the result point set obtained by evaluating f for a subset of the objects in $\mathcal{O}$.

Overall, given a budget η on the number of tuples in $\{\mathcal{R}_i\}_{i \in \mathcal{O}^+ \cup \mathcal{O}^-}$, we first select $\mathcal{O}^+$ with budget η^+, and then use whatever is left of the budget to select $\mathcal{O}^-$.

5.1 Selecting $\mathcal{O}^+$

The objects selection step in Algorithm 2 (especially the choice of $\mathcal{O}^+$) is crucial to the efficiency of the algorithm and to the quality of approximation ($\tilde{\mathbb{S}}_{\text{sparse}}$, $\mathbb{S}_{\text{sketch}}$) to the actual solution ($\mathbb{S}_{\text{sparse}}$, $\mathbb{S} \setminus \mathbb{S}_{\text{sparse}}$). A large $\mathcal{O}^+$ will give $\tilde{\mathbb{S}}_{\text{sparse}}$ a good coverage of $\mathbb{S}_{\text{sparse}}$. However, a large $\mathcal{O}^+$ also leads to slower evaluation. Given a fixed budget η^+ on the number of tuples allowed to be accessed, efficiency would not be an issue anymore, as the number of tuples to be evaluated is limited, so the question becomes how to choose $\mathcal{O}^+$

of limited size in order to maximize the recall of $\tilde{\mathbb{S}}_{\text{sparse}}$ with respect to $\mathbb{S}_{\text{sparse}}$.

Observe that for any point $p \in \mathbb{S}_{\text{sparse}}$ where $p \in f(\mathcal{R}_i)$ (i.e., p comes from object i), it is a necessary condition that $i \in \mathcal{O}^+$ for p to possibly appear in $\tilde{\mathbb{S}}_{\text{sparse}}$. We assume that if an object produces outlier points of $\mathbb{S}$ (when f is evaluated on the full data), it is likely to also produce outlier points when the same query f is evaluated on a sample of its data. Therefore, the prefetched sample serves as a good guide for choosing $\mathcal{O}^+$. We take a partition-based approach to address this problem.

Recall that $\mathbb{S}^{\text{S}}$ denotes the result of executing f on the prefetched sample for all objects (Algorithm 2, line 3). To select objects for $\mathcal{O}^+$, we first partition $\mathbb{S}^{\text{S}}$ into $\mathcal{P}^{\text{S}}$ into grid cells each of size $r_x^{\text{S}} \times r_y^{\text{S}}$. More precisely, $\mathcal{P}^{\text{S}} = \bigcup_{i,j \in \mathbb{Z}} \mathcal{P}_{ij}^{\text{S}}$, where $\mathcal{P}_{ij}^{\text{S}} = \mathbb{S}^{\text{S}} \cap \square_{ij}^{\text{S}}$, $\square_{ij}^{\text{S}} = [i \cdot r_x^{\text{S}}, (i+1) \cdot r_x^{\text{S}}) \times [j \cdot r_y^{\text{S}}, (j+1) \cdot r_y^{\text{S}})$. Based on the partitioning $\mathcal{P}^{\text{S}}$ and the point-object ownership relation, we propose two strategies, namely sparsest-grid-cell (Section 5.1.1) and sparsest-object (Section 5.1.2) for choosing O^+ objects, given an upper bound η^+ on the data size (in number of tuples).

5.1.1 Sparsest Grid Cell

The sparest-grid-cell strategy works as follows. Examine the partitions in non-descending order of their sizes. For a partition $\mathcal{P}_{ij}^{\text{S}}$, include in $\mathcal{O}^+$ all objects that contribute at least one point in $\mathcal{P}_{ij}^{\text{S}}$, i.e., $\mathcal{O}^+ = \mathcal{O}^+ \cup \{k \mid \mathbb{S}_k^{\text{S}} \cap \mathcal{P}_{ij}^{\text{S}} \neq \varnothing\}$. Terminate when the budget η^+ is reached.

We illustrate the idea behind this strategy using the *projection query*. The projection query simply projects the high-dimensional tuples onto a plane defined by two given attributes. Thus, the result on sample $\mathbb{S}^S$ is a random sample of the full result $\mathbb{S}$ of size $\zeta \|\mathcal{D}\|$.

It is known [16, 14] that for a d-dimensional point set P of size n, and a random sample S of P of size $k = (d/\epsilon^2)\log(2n/\delta)$, with probability at least $1 - \delta$, for all d-dimensional axis-aligned rectangle R:

$$\left| \frac{|P \cap R|}{|P|} - \frac{|S \cap R|}{|S|} \right| \leq \epsilon.$$

For projection query, let $P = \mathbb{S}$, $k = \zeta\|\mathcal{D}\|$, $r_x^{\text{S}} = r_x$, and $r_y^{\text{S}} = r_y$, it follows that with probability at least $1 - 2\|\mathcal{D}\| \cdot \exp(\zeta\|\mathcal{D}\|\epsilon^2/2)$, for all $q \in \mathbb{S}^{\text{S}}$:

$$\left| \frac{|\mathcal{N}_{\mathbb{S}}(q; r_x, r_y)|}{\|\mathcal{D}\|} - \frac{|\mathcal{N}_{\mathbb{S}^{\text{S}}}(q; r_x^{\text{S}}, r_y^{\text{S}})|}{\zeta\|\mathcal{D}\|} \right| \leq \epsilon.$$

This means, with high probability, for all points of $\mathbb{S}^{\text{S}}$, its neighborhood density is close to its neighborhood density in $\mathbb{S}$. Therefore, a point of $\mathbb{S}^{\text{S}}$ with a sparser neighborhood has a higher probability of being present in $\mathbb{S}_{\text{sparse}}$.

Similar analysis can be conducted for the other query types as well. While the sparsest-grid-cell strategy is oblivious to the behavior of the query evaluation function f, it follows the idea behind the analysis above by choosing points with sparest neighborhood.

By Corollary 1, for any point p in a partition $\mathcal{P}_{ij}^{\text{S}}$, its number of neighbors $|\mathcal{N}_{\mathbb{S}^{\text{S}}}(p; r_x^{\text{S}}, r_y^{\text{S}})|$ is bounded from below by $|\mathcal{P}_{ij}^{\text{S}}|$. We use this lower bound instead of counting the exact number of neighbors for each point of $\mathbb{S}^{\text{S}}$ for efficiency reasons.

5.1.2 Sparsest Object

An object that contributes to $\mathbb{S}_{\text{sparse}}$ might not be selected by the sparsest-grid-cell strategy due to an "unfortunate" draw of sample. However, if the "overall quality" of the object is good, we can hope to reduce the role of luck in this process by considering the overall sparsity of points produced by this object in $\mathbb{S}^{\text{S}}$.

For each object i, we define its overall sparsity as

$$\mu_i = \mathsf{mean}_{p \in \mathbb{S}^{\text{S}} \cap f(\mathcal{R}_i^{\text{S}})} \{|\mathcal{P}_{ij}^{\text{S}}| \mid p \in \mathcal{P}_{ij}^{\text{S}}\}.$$

In other words, for each object i, we consider the mean neighborhood sparsity of all points in $\mathbb{S}^{\text{S}}$ that are produced by object i. Again, the partition size is used as an approximation for the actual number of neighbors for efficiency reasons.

In fact, the sparsest-grid-cell strategy can be considered as a special case of the sparsest-object strategy with $p = -\infty$ for the power mean function

$$\mathcal{M}_p(x_1, x_2, ...x_n) = \left(\frac{1}{n} \sum_{i=1}^{n} x_i^p \right)^{1/p} \tag{5}$$

It is known that $\mathcal{M}_{-\infty}(x_1, x_2, ..., x_n) = \min(x_1, x_2, ..., x_n)$, representing the strategy deployed by sparsest-grid-cell.

We experiment this strategy with three other instantiations of the power mean function $\mathcal{M}$, namely *arithmetic mean* ($p = 1$), *geometric mean* ($p = 0$), and *harmonic mean* ($p = -1$).

5.2 Selecting $\mathcal{O}^-$

We adopt a simple strategy for selecting the object set $\mathcal{O}^-$—given $\mathcal{O}^+$, include each object of $\mathcal{O} \setminus \mathcal{O}^+$ independently with probability p. The multiplier in Algorithm 2 is set to $\lambda = 1/p$ to maintain expectation, i.e. for any point in $\mathbb{S} \setminus \mathbb{S}^+$, its expected frequency in $\mathbb{S}^-$ is $\lambda \cdot p = 1$.

Another way to think of this strategy is to sample points from $\mathbb{S}$ with correlation. For any object $i \in \mathcal{O} \setminus \mathcal{O}^+$, the sample point set $\mathbb{S}^-$ either includes all points of $f(\mathcal{R}_i)$, or none of it.

$\mathcal{O}^-$ affects the quality of output by Algorithm 2 in several ways. First, not all points of $\mathbb{S}^+$ lead to points in $\mathbb{S}_{\text{sparse}}$ after evaluating f on $\mathcal{O}^+$. It is up to $\mathcal{O}^-$ and $\mathbb{S}^-$ to exclude false positives and include true positives (line 15). Second, the quality of $\mathbb{S}_{\text{sketch}}$ is determined primarily by $\mathbb{S}^-$ (line 20).

5.2.1 Budget Constraint

Note that we need to comply with the total budget constraint η. Let $X_i \sim Ber(p)$ be the Bernoulli random variable denoting if object i is chosen to be included in $\mathcal{O}^-$, for $i \in \mathcal{O} \setminus \mathcal{O}^+$. All X_i's are independent of each other. Let Y be the total number of tuples for objects in $\mathcal{O}^-$. Let $\|\mathcal{O}^+\|$ denote the number of tuples for objects in $\mathcal{O}^+$. Following notations from Section 2.1, the expected number of tuples for $\mathcal{O}^-$ is given by

$$\mu_Y = \mathbb{E}\left[\sum_{i \in \mathcal{O} \setminus \mathcal{O}^+} X_i n_i \right] = \sum_{i \in \mathcal{O} \setminus \mathcal{O}^+} \mathbb{E}[X_i] n_i = p \cdot (\|\mathcal{D}\| - \|\mathcal{O}^+\|).$$

The variance in the number of tuples in $\mathcal{O}^-$ is given by

$$\sigma_Y^2 = \text{Var}\left[\sum_{i \in \mathcal{O} \setminus \mathcal{O}^+} X_i n_i \right] = p(1-p) \cdot \sum_{i \in \mathcal{O} \setminus \mathcal{O}^+} n_i^2.$$

By (one-sided) Chebyshev's inequality, we have

$$\mathbf{Pr}\left[Y \geq (1+\Delta) \cdot \mu_Y\right] \leq \frac{1}{1 + (\Delta\mu_Y/\sigma_Y)^2}.$$

By setting $(1+\Delta)\mu_Y = \eta \cdot \|\mathcal{D}\| - \|\mathcal{O}^+\|$, we have

$$\mathbf{Pr}\left[Y + \|\mathcal{O}^+\| \geq \eta \cdot \|\mathcal{D}\|\right] \leq \frac{1}{1 + (\Delta\mu_Y/\sigma_Y)^2},$$

where

$$\Delta = \frac{\eta \cdot \|\mathcal{D}\| - \|\mathcal{O}^+\|}{\mu_Y} - 1.$$

Since μ_Y is monotone in p, choosing a smaller value of p gives a better chance of complying with the budget constraint η.

5.2.2 *Quality of* $\tilde{\mathbb{S}}_{\text{sparse}}$

We study how $\mathcal{O}^-$ affects the quality of $\tilde{\mathbb{S}}_{\text{sparse}}$ in two ways.

1. For a point in $\mathbb{S}^+ \cap \mathbb{S}_{\text{sparse}}$, what is the minimum probability that it is included in $\tilde{\mathbb{S}}_{\text{sparse}}$?

2. For a point in $\mathbb{S}^+ \setminus \mathbb{S}_{\text{sparse}}$, what is the maximum probability that it is included in $\tilde{\mathbb{S}}_{\text{sparse}}$?

For a point $q \in \mathbb{S}^+ \cap f(\mathcal{R}_i)$, i.e., a point of $\mathbb{S}^+$ coming from object i, let $C_j = |\mathcal{N}_{\mathbb{S}}(q) \cap f(\mathcal{R}_j)|$ be the number of neighbor of q in $\mathbb{S}$ coming from object $j \neq i$. We attempt to provide a bound for the two questions above in the worst case, where all q have at least one neighbor in $\mathbb{S}^+$. Let random variable Z denote the estimated number of q's neighbors. Following the notation of random variable X_i's from Section 5.2.1, Z can be written as follows:

$$Z = 1 + \lambda \cdot \sum_{j \neq i} X_j C_j.$$

And we have

$$\mu_Z = 1 + \sum_{j \neq i} C_j = |\mathcal{N}_{\mathbb{S}}(q)|, \quad \sigma_Z^2 = \frac{1-p}{p} \cdot \sum_{j \neq i} C_j^2.$$

1. If $\mu_Z \leq \tau$, by Chebyshev's inequality, we have

$$\mathbf{Pr}[Z > \tau] \leq \frac{1}{1 + (\mu_Z - \tau)^2/\sigma_Z^2}.$$

2. If $\mu_Z > \tau$, symmetrically, we have

$$\mathbf{Pr}[Z \leq \tau] \leq \frac{1}{1 + (\mu_Z - \tau)^2/\sigma_Z^2}.$$

These bounds suggest a couple of things. First, the farther μ_Z deviates from τ, the more confident we can be that Algorithm 2 will make the right decision on whether to include q in $\tilde{\mathbb{S}}_{\text{sparse}}$. In other words, it is harder to classify points correctly whose actual neighborhood density in $\mathbb{S}$ is close to the sparsity threshold τ. Also, since τ is presumably small, in general it is harder to classify points of $\mathbb{S}_{\text{sparse}}$ correctly than $\mathbb{S} \setminus \mathbb{S}_{\text{sparse}}$, i.e., high recall is harder to achieve than high precision. Second, a larger budget for $\mathcal{O}^-$ would lead to a larger p, thus higher confidence in classification.

5.2.3 *Quality of* $\mathbb{S}_{\text{sketch}}$

Recall that the sketch distance (defined in Section 2.2) is used to measure the quality of a sketch $\mathbb{S}_{\text{sketch}}$ w.r.t. $\mathbb{S} \setminus \mathbb{S}_{\text{sparse}}$. While the exact sketch distance is not easy to come by, an upper bound can be obtained as follows.

Ignore $\mathbb{S}_{\text{sparse}}$ for now. Partition the result set $\mathbb{S}$ such that any two points in the same partition are neighbors. For example, under L_∞-norm neighborhood definition, partition $\mathbb{S}$ into grid cells each of size $r_x \times r_y$. Let $P_1, \ldots, P_m$ denote the resulting partitions. Let $Z_j = \lambda \cdot |\mathbb{S} \cap P_j|$ be the estimated number of points in partition P_j by the sketch. Let $Z = \sum_j Z_j$ be the estimated total number of points.

We have the following bound on $\delta(\mathbb{S}, \mathbb{S}_{\text{sketch}})$, expressed in terms of the means and variances of Z_j's and Z.

$$\mathbf{Pr}\left[\delta(\mathbb{S}, \mathbb{S}_{\text{sketch}}) \geq 1 - \frac{1-\Delta^-}{1+\Delta^+}\right] \leq \sum_j \frac{1}{1 + (\Delta^- \mu_j/\sigma_j)^2} + \frac{1}{1 + (\Delta^+ \mu_Z/\sigma_Z)^2}. \quad (6)$$

Tighter bounds can be obtained by taking into account the correlation among Z_j's. When the distributions of number of points by objects are identical in all partitions, we have

$$\mathbf{Pr}\left[\delta(\mathbb{S}, \mathbb{S}_{\text{sketch}}) \geq 1 - \frac{1-\Delta}{1+\Delta}\right] \leq \frac{2}{1 + (\Delta \mu_Z/\sigma_Z)^2}. \quad (7)$$

If, on top of the identical distribution scenario, C_i's are all equal, we have $\frac{\mu_Z^2}{\sigma_Z^2} = \frac{Np}{1-p}$, and

$$\mathbf{Pr}\left[\delta(\mathbb{S}, \mathbb{S}_{\text{sketch}}) \geq 1 - \frac{1-\Delta}{1+\Delta}\right] \leq \frac{2}{1 + \Delta^2 Np/(1-p)}. \quad (8)$$

On the other hand, in the worse case where all Z_j's are independent and $\mu_j = \sigma_j$, we would have

$$\mathbf{Pr}\left[\delta(\mathbb{S}, \mathbb{S}_{\text{sketch}}) \geq 1 - \frac{1-\Delta}{1+\Delta}\right] \leq \frac{N+1}{1 + \Delta^2 p/(1-p)}. \quad (9)$$

Taking $\mathbb{S}_{\text{sparse}}$ back into account, since the frequency counts in $\mathbb{S}^+$ are all precise (estimates with zero variance), including $\mathbb{S}_{\text{sparse}}$ does not nullify any results above.

Proofs of the above bounds and additional remarks can be found in the appendix. Note that these bounds may still be quite loose. We will show the quality of sketch produced by Algorithm 2 via empirical results in Section 6.

6 Experiments

We implemented our algorithms in C++ and evaluated the efficiency and result quality on three real datasets. All experiments are conducted on a machine with Intel Core i7-2600 3.4GHz CPU and 7.8GB RAM.

6.1 Datasets

NBA players' game-by-game performance statistics (NBA)[7]: By considering a player as an object and his performance stats in each game as a tuple, the dataset consists of 1.01 million tuples ($\|\mathcal{D}\| \approx 1.01 \times 10^6$) for a total of 3,129 players ($n = 3129$).

DBLP[8]: For the DBLP data set, we consider authors as objects, and year-by-year performance of an author in terms of the author's numbers of publications in different venues; i.e., a tuple represents an author's performance in a given year, where attributes represent the author's publication counts in different venues in that year. This dataset consists of ∼67k tuples ($\|\mathcal{D}\| \approx 67 \times 10^3$) for ∼32k distinct authors ($n \approx 32 \times 10^3$)

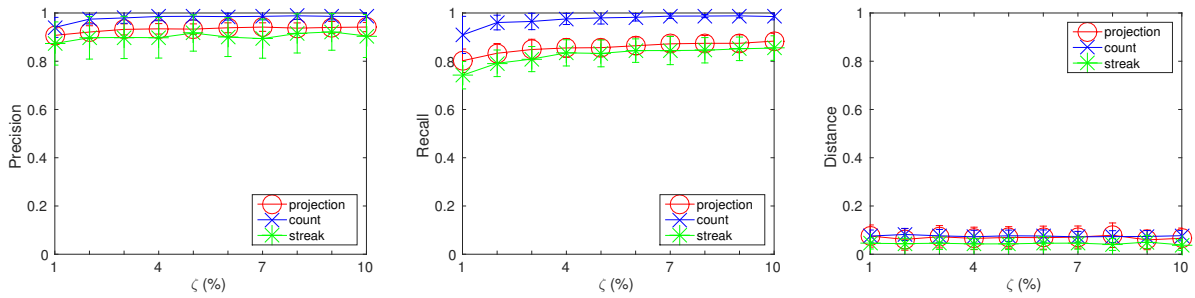
Wikipedia edit history data (WIKI)[9]: The raw Wikipedia edit log in the main namespace consists of a total of ∼116.6 million entries, each representing a revision of an article, with information of the user ID, article category, edit size, etc. We consider users as objects, and a tuple is the (number, size) of all edits and minor edits in one day. For over 3.3 million users ($n \approx 3.3 \times 10^6$), we have a total of ∼8.91 million tuples ($\|\mathcal{D}\| \approx 8.91 \times 10^6$).

6.2 Quality Evaluation

We evaluated the quality of $(\tilde{\mathbb{S}}_{\text{sparse}}, \mathbb{S}_{\text{sketch}})$ in two aspects—(i) quality of $\tilde{\mathbb{S}}_{\text{sparse}}$ w.r.t. $\mathbb{S}_{\text{sparse}}$ in terms of recall and, less importantly, precision, and (ii) quality of $\mathbb{S}_{\text{sketch}}$ w.r.t. $\mathbb{S} \setminus \mathbb{S}_{\text{sketch}}$ as measured by the sketch distance function δ.

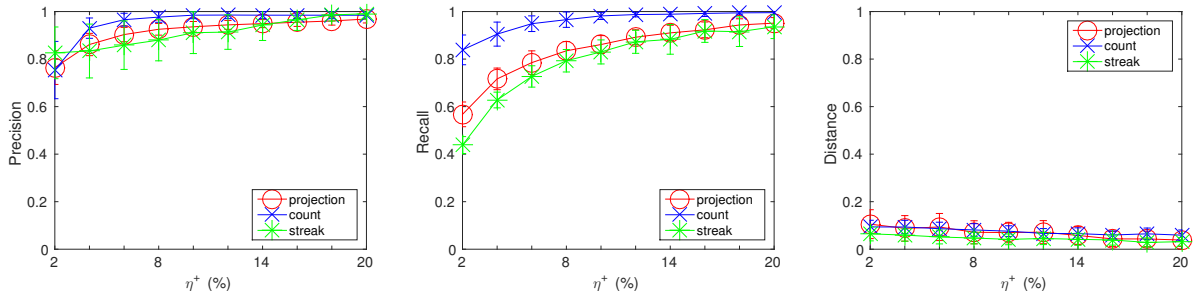
Note that the baseline algorithm (Algorithm 1) performs full evaluation, and hence produces the exact $\mathbb{S}_{\text{sparse}}$. Also, under L_∞-norm (rectangular neighborhood), the grid-partition-based baseline algorithm trivially achieves $\delta(\mathbb{S}_{\text{sketch}}, \mathbb{S} \setminus \mathbb{S}_{\text{sparse}}) = 0$. Therefore,

[7] http://www.basketball-reference.com
[8] http://dblp.uni-trier.de
[9] https://snap.stanford.edu/data/wiki-meta.html



(a) Precision (b) Recall (c) Sketch Distance

Figure 4: **(NBA)** Vary sample rate ζ between 1-10%. Fix $\eta^+ = \eta^- = .1$.



(a) Precision (b) Recall (c) Sketch Distance

Figure 5: **(NBA)** Vary budget η^+ for $\mathcal{O}^+$ between 2-20%. Fix $\zeta = .05$, $\eta^- = .1$.

we are only interested in evaluating the quality of results of the sampling-based algorithm (Algorithm 2).

The same set of experiments were conducted on all three datasets. We first illustrate observations common to all data sets using the NBA data set.

Impact of $\mathcal{O}^+$ selection strategy and neighborhood definition. We tested the performance of Algorithm 2 with (i) different strategies for selecting $\mathcal{O}^+$ described in Section 5.1 ($\mathcal{M}_p$ with $p = 1$, 0, -1, $-\infty$), and (ii) different $\|\cdot\|$ functions in the neighborhood definition (Eq. 2.1), namely L_1-, L_2-, and L_∞-norms, corresponding to diamondoid, oval, and rectangular neighborhood, respectively. No significant difference was observed in the quality of result produced by Algorithm 2. We omit the figures here due to limited space.

Varying sample rate ζ. Performance of Algorithm 2 at different sample rates ζ is shown in Figure 4. Results suggest that increasing ζ gives notable improvement in the quality of $\tilde{\mathcal{S}}_{\text{sparse}}$ in the beginning (Figures 4a and 4b), but the improvement diminishes afterwards (beyond 5% for NBA). On the other hand, ζ has almost no impact on the quality of the sketch $\mathcal{S}_{\text{sketch}}$ in terms of distance to the exact solution $\mathcal{S} \setminus \mathcal{S}_{\text{sparse}}$ (Figure 4c).

Varying budget η^+ for $\mathcal{O}^+$. In Figure 5, we fix ζ and η^-, and observe the performance of Algorithm 2 with varying budget η^+. We see that as η^+ increases, precision and recall keep increasing and approach 100% (Figures 5a and 5b). Similar to the results for varying ζ, η^+ shows little impact on the quality of $\mathcal{S}_{\text{sketch}}$ (Figure 5c).

Varying budget η^- for $\mathcal{O}^-$. In contrast to ζ and η^+, increasing η^- gives big improvement on the quality of $\mathcal{S}_{\text{sketch}}$ (Figure 6c), while recall of $\tilde{\mathcal{S}}_{\text{sparse}}$ is barely affected (Figure 5b). It is worth noting that having too few objects $\mathcal{O}^-$ may increase the amount of false positives in $\tilde{\mathcal{S}}_{\text{sparse}}$, accounting for the increase in precision shown in Figure 5a.

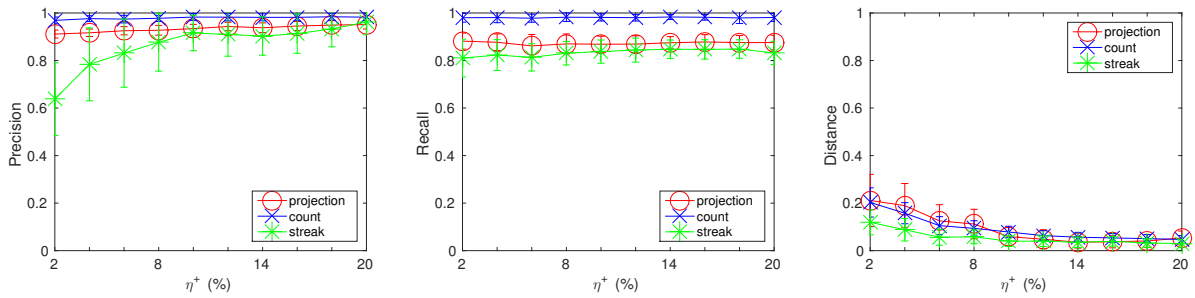
Varying overall budget η. Results above show how η^+ or η^-, when the other is fixed, affects different aspects of result quality. Now we show the tradeoff between η^+ and η^- when fixing the overall budget η. In Figure 7, we fix the total budget $\eta = \eta^+ + \eta^- = 20\%$, and show the three quality indicators, varying the budget allocation between η^+ and η^-. From the earlier experiments, it is expected that a larger η^+ would lead to better precision and recall, while a larger η^- would lead to a better sketch with a smaller distance to the ground truth.

To measure the overall quality of a solution by Algorithm 2, we use a generalized version of the F-score, by taking the weighted harmonic mean of the three quality measure, precision, recall, and distance. We assign weights β_P^2, β_R^2, β_δ^2 to precision, recall, and $1-\delta$, respectively, where $\beta_R > \beta_\delta > \beta_P > 0$, signifying decreasing importance. Formally,

$$F_{\beta_P,\beta_R,\beta_\delta} = \frac{\beta_P^2 + \beta_R^2 + \beta_\delta^2}{\frac{\beta_P^2}{P} + \frac{\beta_R^2}{R} + \frac{\beta_\delta^2}{1-\delta(\mathcal{S}_{\text{sketch}}, \mathcal{S} \setminus \mathcal{S}_{\text{sparse}})}}. \tag{10}$$

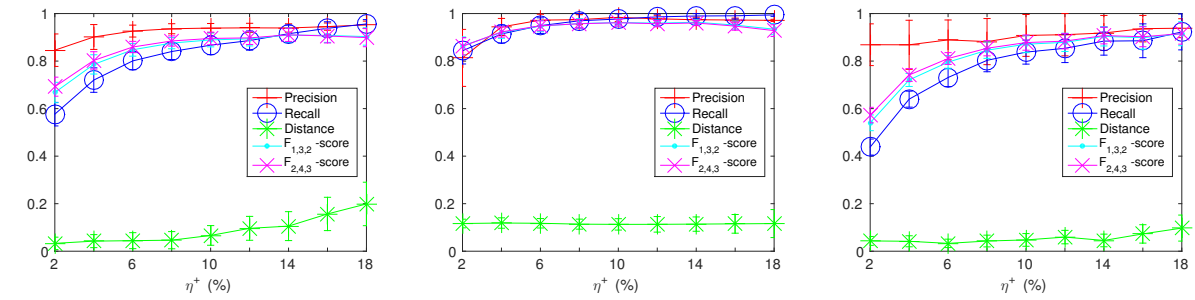
We show results of two F-scores, $F_{1,3,2}$ and $F_{2,4,3}$. In Figure 7a and 7b, both $F_{1,3,2}$ and $F_{2,4,3}$ peak at $\eta^+ = 14\%$, leaving η^- at 6%. On the contrary, for streak query (Figure 7c), the quality of $\mathcal{S}_{\text{sketch}}$ is still decent even with η^- as small as 2%, which allows the majority of the overall budget to be allocated to $\mathcal{O}^+$ in order to improve the quality of $\tilde{\mathcal{S}}_{\text{sparse}}$.

Precision versus recall. From the results on NBA, we see a higher precision than recall when sufficient budget is given. The same holds for results on DBLP and WIKI. As we have shown in Section 5.2.2, $\mathcal{S}^-$ plays an important role in both eliminating false positives from and keeping true positives in $\mathcal{S}^+$. While the probability of false positives only depends on the rate at which objects are sampled from $\mathcal{O} \setminus \mathcal{O}^+$ to form $\mathcal{O}^-$, the probability of false negatives is conditional on the fact that the object that yields some point of $\mathcal{S}^+ \cap \tilde{\mathcal{S}}_{\text{sparse}}$ is included in $\mathcal{O}^+$ in the first place. And we discuss next why it is hard to ensure the right objects are chosen for $\mathcal{O}^+$.



(a) Precision (b) Recall (c) Sketch Distance

Figure 6: **(NBA)** Vary budget η^- for $\mathcal{O}^-$ between 2-20%. Fix $\zeta = .05$, $\eta^+ = .1$.



(a) Projection (b) Count (c) Streak

Figure 7: **(NBA)** Fix total budget $\eta = .2$ for $\mathcal{O}^+$ and $\mathcal{O}^-$, vary η^+ for $\mathcal{O}^+$ between 2-18%.

We should note that the effectiveness of $\mathcal{O}^+$ selection depends on both the data distributions and the query type. Algorithm 2 assumes neither. To see how the data distribution affects the effectiveness of $\mathcal{O}^+$, we compare the recall of $\tilde{\mathcal{S}}_{\text{sparse}}$ on NBA and on WIKI. In Figure 8a, we show the recall of $\tilde{\mathcal{S}}_{\text{sparse}}$ on WIKI data with varying sample rate. The result is significantly worse than that on the NBA data (Figure 4b). For NBA, the average number of tuples per player is over 300. In contrast, the average number of tuples per author is below 2. Even the most renowned researchers have tuples for fewer than 40 years. The consequence of a small sample rate is a high probability of missing all tuples of an author, even for the most prolific ones. A second set of results shown in Figure 8a is based on a different sampling method: sample for each object at a rate proportional to the size of the object, while the overall sample rate remains the same. By adopting this sampling strategy, we implicitly assume that points of $\mathcal{S}_{\text{sparse}}$ are likely to be yielded by objects with more tuples, and we get significant improvement over uniform sampling.

The other important factor that influences the effectiveness of $\mathcal{O}^+$ selection is the query type. On the NBA data, we see a notable difference between the recall $\tilde{\mathcal{S}}_{\text{sparse}}$ for count query and the other two query types. Similarly, on the WIKI data, we observe a notable difference between *streak* query and the other two types (Figure 9). For *streak* query, a point of $\mathcal{S}_{\text{sparse}}$ comes from a (possibly small) number of consecutive tuples. Uniformly sampled tuples may not be representative at all for the overall quality of an object w.r.t. a streak query. Similarly, for *projection* query, say for NBA, a generally lousy player could have one outstanding game that contributes to $\mathcal{S}_{\text{sparse}}$, which will unlikely be picked up by a small uniform sample. On the contrary, any point in $\mathcal{S}$ of a *count* query considers all tuples of an object, making $\mathcal{O}^+$ selection less vulnerable to uniform samples. Had Algorithm 2 known the behavior of the query function f, smarter sampling strategies could be deployed to better guide the selection of $\mathcal{O}^+$.

6.3 Efficiency

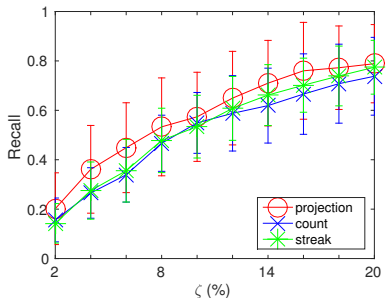
We evaluate the efficiency of the sampling-based algorithm when varying various parameters, namely (i) the object selection strategy, (ii) the distance metric (L_∞-/L_1-/L_2-norm), and (iii) the computation budget η. The efficiency is measured as execution-time speed-up over the baseline algorithm (Algorithm 1), which performs full evaluation on all objects. Not surprisingly, the object selection strategy and the distance metric do not influence the efficiency of Algorithm 2. Hence, the results are omitted.

Varying ζ, η^+, and η^-. Figure 10 shows the speed-up of the sampling-based algorithm over the baseline, varying ζ, η^+, and η^-, respectively, with the other two fixed. The solid *optimal* curve shows the maximum possible speed-up, i.e. $(\zeta + \eta^+ + \eta^-)^{-1}$, for linear-time query function f, excluding any overhead during object selection. Roughly the same amount of speed-up is observed for all three query types, regardless of the size and distribution of the input data. Results on DBLP and WIKI data are similar and thus omitted due to limited space.

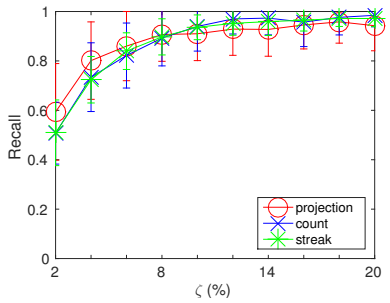
7 Related Works and Discussion

Visualization-assisted data analysis has recently attracted a lot of attention in both database and CHI communities. Jugel et al. [11] studied data reduction techniques for high-volume time series data driven by visualization constraints, while our work is motivated by limit on data access. Along the same line, the idea of data reduction for efficient visualization of time series data had previously been explored by Burtini et al. [3].

The problem of rapid generation of approximate visualizations while preserving crucial properties was studied by Blais et al. [13] for bar chart, which employs very different techniques. Efficiency-precision tradeoff for exploratory visualization of large dataset has also been studied in the CHI community [8, 9]. The idea of using prefetching techniques to improve visualization efficiency for



(a) Uniform sample



(b) Sample rate proportional to objects' data size

Figure 8: **(DBLP)** Compare recall of $\tilde{\mathcal{S}}_{\text{sparse}}$ with two different sampling strategies, at varying sample rate 1-10%, fixing $\eta^+ = \eta^- = .1$.

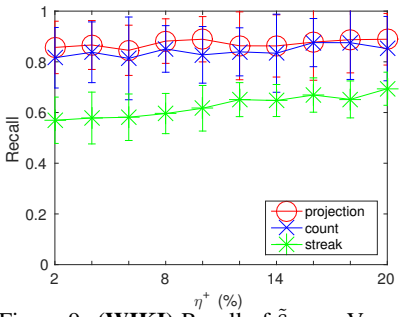
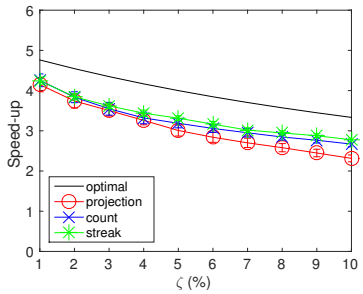
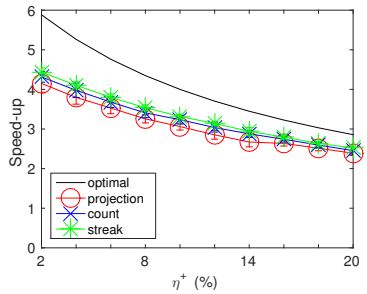


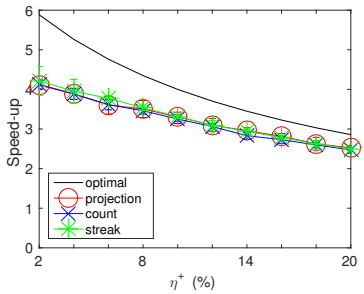
Figure 9: **(WIKI)** Recall of $\tilde{\mathcal{S}}_{\text{sparse}}$. Vary η^+ 2-20%. Fix $\zeta = .05, \eta^- = .1$



(a) Vary ζ. Fix $\eta^+ = \eta^- = .1$.



(b) Vary η^+. Fix $\zeta = .05, \eta^- = .1$.



(c) Vary η^-. Fix $\zeta = .05, \eta^+ = .1$.

Figure 10: **(NBA)** Efficiency of algorithms varying sample rate and budget.

real-time responses in an interactive environment has been studied by Doshi et al. [7]. Kandel et al. presented *Profiler* [12] for visualization-assisted data exploration and anomaly detection from an HCI perspective.

In this paper, we have considered outliers as points having a small number of neighbors (thus not necessarily on the skyline), but the idea of finding global outliers from promising objects by evaluating sample data can apply to outliers of other forms, such as skyline points [2]. The sketching algorithm using points of $\mathcal{S}^+ \setminus \tilde{\mathcal{S}}_{\text{sparse}}$ along with $\mathcal{S}^-$ with estimated counts can work as is.

The type of uniqueness-based fact-finding has been studied for skyline and skyband points [17]. There, specialized algorithms were proposed to efficiently find all interesting and unique points of a large point set; in this paper, we propose a neighborhood sparsity based uniqueness definition and propose an algorithm tailoring towards the visualization method instead of the claim type.

The idea behind the two-phase sampling-based algorithm is related to the notion of *coreset* [1] in computational geometry. Since we do not have direct access to the result set $\mathcal{S}$, for each query instance f, we construct "coresets" of objects $\mathcal{O}^+$ and $\mathcal{O}^-$ instead of coresets of points. The selection of $\mathcal{O}^-$ using random sampling is analogous to drawing random samples from the point set [16, 14]. On the other hand, the selection of $\mathcal{O}^+$ is optimized towards preserving another extent measure: minimum density points of $\mathcal{S}$.

Experiments in this paper are all conducted in memory. However, at service time, to support exploration query evaluation for many datasets, and to avoid dedicating the memory to a single dataset, objects' full data is hosted in SSTable [4] and brought into memory only upon request via the SSTable service API. Data accessing cost will become even more dominant in the execution time. In that case, our sampling-based algorithm with a data access budget would increase its advantage over the baseline algorithm. Also, we have not explored the possibility of parallel evaluation of query function f. On large data sets, parallel query evaluation for different objects will further speed up the overall efficiency.

8 Conclusion

In journalism, claims derived from data are important ingredients for many stories, ranging from politics to sports. A common analysis for determining the quality of a claim is to compare it with other claims of the same form. Such exploratory analysis can usually be carried out effectively via visualization. In this paper, we consider claims that can be modeled as queries whose results can be represented as 2d points, and we focus on one common type of visualization—a combination of 2d scatter plot for outliers and a heatmap for overall distribution. We propose an efficient two-phase sampling-based algorithm that works with any query function. The algorithm first executes the query function on a sample of the dataset, and then, based on the result, further selects additional data to access in order to produce a final approximate answer. Experiments show that our algorithm is efficient and is able to preserve result properties important to visualization—namely the outliers and the overall distribution.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and Computational Geometry*, 52:1–30, 2005.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 2001 International Conference on Data Engineering*, pages 421–430, Heidelberg, Germany, Apr. 2001.

[3] G. Burtini, S. Fazackerley, and R. Lawrence. Time series compression for adaptive chart generation. In *Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE*

Canadian Conference on, pages 1–6. IEEE, 2013.

[4] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 26(2):4, 2008.

[5] S. Cohen, J. T. Hamilton, and F. Turner. Computational journalism. *Communications of the ACM*, 54(10):66–71, 2011.

[6] S. Cohen, C. Li, J. Yang, and C. Yu. Computational journalism: A call to arms to database researchers. In *Proceedings of the 2011 Conference on Innovative Data Systems Research*, Asilomar, California, USA, Jan. 2011.

[7] P. R. Doshi, E. A. Rundensteiner, and M. O. Ward. Prefetching for visual data exploration. In *Proceedings of the 2003 International Conference on Database Systems for Advanced Applications*, pages 195–202, Kyoto, Japan, Mar. 2003. IEEE.

[8] D. Fisher. Incremental, approximate database queries and uncertainty for exploratory visualization. In *Proceedings of the 2011 Large Data Analysis and Visualization*, pages 73–80, Paris, France, Nov. 2011. IEEE.

[9] D. Fisher, I. Popov, S. Drucker, and mc schraefel. Trust me, i'm partially right: Incremental visualization lets analysts explore large datasets faster. In *Proceedings of the 2012 International Conference on Human Factors in Computing Systems*, pages 1673–1682, Austin, Texas, USA, May 2012. ACM.

[10] X. Jiang, C. Li, P. Luo, M. Wang, and Y. Yu. Prominent streak discovery in sequence data. In *Proceedings of the 2011 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1280–1288, San Diego, California, USA, Aug. 2011.

[11] U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: a visualization-oriented time series data aggregation. *Proceedings of the VLDB Endowment*, 7(10):797–808, 2014.

[12] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, and J. Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Proceedings of the 2012 International Conference on Advanced Visual Interfaces*, pages 547–554, Capri Island, Naples, Italy, May 2012. ACM.

[13] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *arXiv preprint arXiv:1412.3040*, 2014.

[14] J. M. Phillips. Chernoff-hoeffding inequality and applications. *arXiv preprint arXiv:1209.6396*, 2012.

[15] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 6th International Conference onComputer Vision*, pages 59–66. IEEE, 2008.

[16] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16(2):264–280, 1971.

[17] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. On "one of the few" objects. In *Proceedings of the 2012 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495, Beijing, China, Aug. 2012.

APPENDIX

Following the notations of Section 5.2.3, we derive the bounds on $\delta(\mathcal{S}, \mathcal{S}_{\text{sketch}})$ here.

First, by linearity of expectation, we have

$$\mu_j = \mathbb{E}[Z_j] = |\mathcal{S} \cap P_j|,\ \mu_Z = \mathbb{E}[Z] = |\mathcal{S}|.$$

It is easy to see the following bound on $\delta(\mathcal{S}, \mathcal{S}_{\text{sketch}})$:

$$\delta(\mathcal{S}, \mathcal{S}_{\text{sketch}}) \le 1 - \frac{\sum_j \min\{Z_j, |\mathcal{S} \cap P_j|\}}{\max\{Z, |\mathcal{S}|\}} = 1 - \frac{\sum_j \min\{Z_j, \mu_j\}}{\max\{Z, \mu_Z\}}.$$

By the (one-sided) Chebyshev's inequality, we have

$$\begin{aligned}
&\mathbf{Pr}\left[\sum_j \min\{Z_j, \mu_j\} \ge (1-\Delta^-)\mu_Z\right] \\
\ge &\mathbf{Pr}\left[\bigcap_j Z_j \ge (1-\Delta^-)\mu_j\right] \\
\ge &1 - \mathbf{Pr}\left[\bigcup_j Z_j \le (1-\Delta^-)\mu_j\right] \\
\ge &1 - \sum_j \frac{1}{1 + (\Delta^-\mu_j/\sigma_j)^2}
\end{aligned}$$

and

$$\begin{aligned}
\mathbf{Pr}\left[\max\{Z, \mu_Z\} \ge (1+\Delta^+)\mu_Z\right] &= \mathbf{Pr}\left[Z \ge (1+\Delta^+)\mu_Z\right] \\
&\le \frac{1}{1 + (\Delta^+\mu_Z/\sigma_Z)^2}.
\end{aligned}$$

Combining the two inequalities above, we have

$$\begin{aligned}
&\mathbf{Pr}\left[\delta(\mathcal{S}, \mathcal{S}_{\text{sketch}}) \ge 1 - \frac{1-\Delta^-}{1+\Delta^+}\right] \\
\le &\mathbf{Pr}\left[\sum_j \min\{Z_j, \mu_j\} \le (1-\Delta^-)\mu_Z \vee Z \ge (1+\Delta^-)\mu_Z\right] \\
\le &\sum_j \frac{1}{1 + (\Delta^-\mu_j/\sigma_j)^2} + \frac{1}{1 + (\Delta^+\mu_Z/\sigma_Z)^2}.
\end{aligned}$$

To see the bounds dependent on correlation of point distribution by objects, let c_{ij} denote the number of points in partition P_j that come from object i; i.e., $c_{ij} = |f(\mathcal{R}_i) \cap P_j|$, and $C_i = \sum_j c_{ij} = |f(\mathcal{R}_i)|$. It follows that $Z_j = \lambda \cdot \sum_i X_i c_{ij}$. We have, for each i:

$$\mu_i = \sum_j c_{ij}, \quad \sigma_i^2 = \frac{1-p}{p} \cdot \sum_i c_{ij}^2,$$

and for Z,

$$\mu_Z = \sum C_j, \quad \sigma_Z^2 = \frac{1-p}{p} \cdot \sum_i C_i^2.$$

For any two partitions j and j', the covariance and correlation between Z_j and $Z_{j'}$ are given by

$$\sigma_{jj'} = \frac{1-p}{p} \cdot \sum_i c_{ij} c_{ij'}, \quad \rho_{jj'} = \frac{\sigma_{jj'}}{\sigma_j \sigma_{j'}}.$$

The tighter bound can be obtained by taking into account the correlation among Z_j's, using the dependent multi-variate Chebyshev's inequality as follows:

$$\begin{aligned}
&\mathbf{Pr}\left[\bigcap_j Z_j \ge (1-\Delta^-)\mu_j\right] \\
\le &1 - \frac{1}{m^2}\left(\sqrt{u} + \sqrt{m-1}\sqrt{\frac{m}{(\Delta^-)^2}\sum_j \frac{\sigma_j^2}{\mu_j^2} - u}\right)^2,
\end{aligned}$$

where

$$u = \frac{1}{(\Delta^-)^2} \sum_j \sum_{j'} \frac{\rho_{jj'}}{\mu_j \mu_{j'}}.$$

Setting $\rho_{jj'} = 1$ for all j, j', and we get Ineq. (7), (8), and (9).