Exploiting Visual Perception for Sampling-Based Approximation on Aggregate Queries

Daniel Alabi*

Columbia University

Abstract

Efficient sampling algorithms have been developed for approximating answers to aggregate queries on large data sets. In some formulations of the problem, concentration inequalities (such as Hoeffding's inequality) are used to estimate the confidence interval for an approximated aggregated value. Samples are usually chosen until the confidence interval is arbitrarily *small enough* regardless of how the approximated query answers will be used (for example, in interactive visualizations). In this report, we show how to exploit visualization-specific properties to reduce the sampling complexity of a sampling-based approximate query processing algorithm while preserving certain visualization guarantees (the visual property of relative ordering) with a very high probability.

1 Introduction

In order to display interactive visualizations faster, Wu & Nandi [1] suggest using invertible *perceptual* and *encoding* functions to approximate query answers. Both types of functions would be specified as part of **visualization specific language extensions**: RENDERED BY and PERCEIVED BY clauses would be used to specify the encoding and perceptual functions respectively. The language extensions are part of an overarching project called **InterVis**, which consists of two major new components: InterVis-CACHE and InterVis-APPROX. Whereas InterVis-CACHE handles session-based caching of queries, InterVis-APPROX facilitates the use of visualization-aware approximation.

Most approximate query processing systems (pivotal examples include [2, 3, 4]) don't take interaction into account when computing query answers. The goal of InterVis is to create a system where every layer is **interaction-aware** to better exploit previously unexplored optimizations in querying, processing, and visualizing the data. One avenue of optimization is to use the user-specified perceptual and encoding functions to approximate query answers in order to limit resource usage for interactive visualizations. Two types of perceptual functions were presented: univariate and bivariate. Univariate perceptual functions of the form $P : \mathbb{R} \to \mathbb{R}$ map a visually encoded value to the perceived error. On the other hand, bivariate perceptual functions of the form $P : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ map a pair of encoded values to the error in the perceived proportional differences. A simple linear perceptual function like $P(v) = 5 \cdot 10^{-5}v$ means that a user can perceive a 10^5 pixel height within a margin of error of ± 5 pixels. For more complex and realistic perceptual functions, according to Stevens' power law of theoretical psychophysics, refer to Section 4.1. On the other hand, a simple encoding function is $E(v) = \lfloor \frac{v}{10^4} \rfloor$, for example, if values in the domain $[0, 10^6]$ are mapped to a maximum bar height of 100 pixels. **How can we use encoding and perceptual functions such as the ones defined above to approximate query answers?**

This report attempts to answer the question by presenting IFocusViz , a sampling-based approximate query processing algorithm that utilizes encoding and perceptual functions to converge faster on the estimated query answers.

2 Approach

For this report, we consider the following abstract query:

`SELECT` X, `AVG`(Y) `FROM` $R(X, Y)$ `GROUP BY` X

This query can be translated to a map visualization such as the one in Figure 2. While we restrict ourselves to queries with a single `GROUP BY` and a `AVG` aggregate, the query processing algorithm presented can be extended to a much more general class of queries and visualizations.

*daniel.alabi@columbia.edu

Data: $S_1, \ldots, S_k, \delta, E, P$
Initialize $m \leftarrow 1$;
Draw sample s_i from each of $S_1, ..., S_k$
to provide initial estimates $v_1, \ldots, v_k$ where $v_i = s_i$;
Initialize $A = \{1, \ldots, k\}$;
while $A \neq \emptyset$ **do**
  $m \leftarrow m + 1$;
  $\epsilon = \sqrt{(1 - \frac{m/2-1}{\max_{i \in A} |S_i|}) \frac{2 \log \log(m) + \log(\pi^2 k / 3\delta)}{2m}}$;
  foreach $i \in A$ **do**
    Draw sample s_i from S_i;
    $v_i = \frac{m-1}{m} v_i + \frac{1}{m}(s_i)$;
  end
  foreach $i \in A$ **do**
    if $[v_i - \epsilon, v_i + \epsilon] \cap (\bigcup_{j \in A \setminus \{i\}} [v_j - \epsilon, v_j + \epsilon]) = \emptyset$ **then**
      A $\leftarrow A \setminus \{i\}$;
    else
      select $j \neq i$ such that $[v_j - \epsilon, v_j + \epsilon]$ overlaps most with $[v_i - \epsilon, v_i + \epsilon]$;
      $e_{i1} = E(v_i - \epsilon)$;
      $e_{i2} = E(v_i + \epsilon)$;
      $e_{j1} = E(v_j - \epsilon)$;
      $e_{j2} = E(v_j + \epsilon)$;
      if $e_{i2} - e_{j1} \leq P(e_{i2}, e_{j1})$ `or` $e_{j2} - e_{i1} \leq P(e_{j2}, e_{i1})$ **then**
        $A \leftarrow A \setminus \{i\}$;
      else
    end
  end
end

Algorithm 1: IFocusviz

k	Number of groups
$S_1, \ldots, S_k$	The k groups themselves.
δ	The confidence intervals of all active groups contains actual averages for the groups with probability greater than $1 - \delta$
E	Encoding function
P	Bivariate perceptual function

Table 1: Table of Notation for Algorithm 1

IFocusViz is an adaptation of the IFocus algorithm presented by Blais et al. [5]. Table 1 describes the symbols used in Algorithm 1.

At a high level, the IFocus algorithm works by maintaining, for each group, a confidence interval within with the algorithm believes the true average of each group lies. The groups whose confidence intervals overlap with other groups are called *active groups*. The algorithm proceeds in rounds. For each round, a single additional sample is taken per active group. The algorithm terminates when there are no remaining active groups. The estimated averages $v_1, \ldots, v_k$ are then returned to the caller. Figure 1 is an illustrative diagram showing intermediate processing of the approximate departure delay times for some U.S. states. Section 3 further describes results from running IFocusViz on a flights data set.

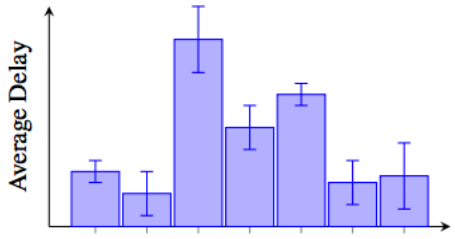

Figure 1: Intermediate Processing of Average Delay Times

IFocusViz is different from IFocus in its determination of active groups. In IFocusViz , *active groups* are groups whose overlap with confidence intervals of other groups is *perceptually discernible*. We use perceptual and encoding functions to determine what overlap is perceptually discernible. An illustrative example of how IFocusViz determines active groups is given below.

Example: *Suppose we have groups* $S_1, \ldots, S_5$ *and* $\delta = 0.05$ *(*IFocusViz *obeys the visual ordering property with probability greater than* 0.95*). Also, assume we are given perceptual and encoding functions of* $P(\cdot, \cdot) = 0$ *(the most conservative perceptual function) and* $E(v) = \lfloor \frac{v}{10} \rfloor$ *respectively. At the* r*th* ($m = r, r \geq 1$ *in Algorithm 1) iteration, with* $\epsilon = 10.0$*, the confidence intervals for groups* $S_1, \ldots, S_5$ *are:*

$$
\begin{aligned}
S_1 &\rightarrow [11, 31] \\
S_2 &\rightarrow [523, 543] \\
S_3 &\rightarrow [603, 623] \\
S_4 &\rightarrow [621, 641] \\
S_5 &\rightarrow [625, 645]
\end{aligned}
$$

Then, in the range of the encoding function, the confidence intervals are:

$$
\begin{aligned}
S_1 &\rightarrow [1, 3] \\
S_2 &\rightarrow [52, 54] \\
S_3 &\rightarrow [60, 62] \\
S_4 &\rightarrow [62, 64] \\
S_5 &\rightarrow [62, 64]
\end{aligned}
$$

At this stage, both IFocus *and* IFocusViz *will identify groups* S_1, S_2 *as non-active since their confidence intervals don't overlap with any other groups. On the other hand, groups* S_4, S_5 *are active since their confidence intervals overlap and the length of the overlap (in the range of the encoding function) is greater than* $P(\cdot, \cdot) = 0$*. As for* S_3*,* IFocus *will identify the group as active since there is a non-empty overlap (*$[62, 62]$*) between the confidence intervals for* S_3 *and* S_4*. But* IFocusViz *will consider group* S_3 *to be non-active since the length of the overlap is less than or equal to the perceptual function* $P(\cdot, \cdot)$ *(which implies that the overlap is not perceptually discernible).*

3 Experiments

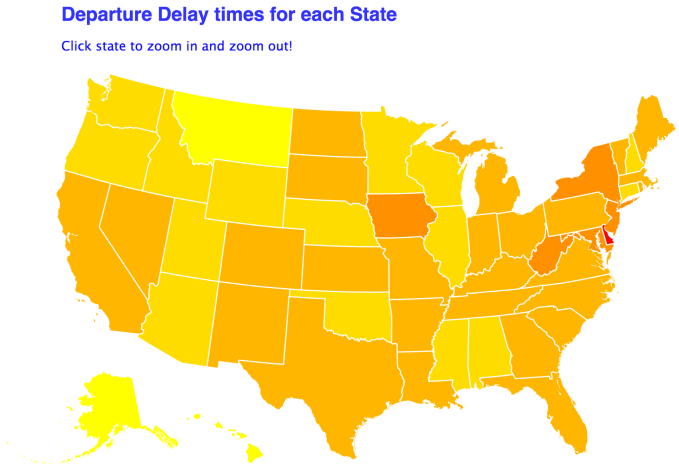

Figure 2: Visualization of approximate average departure delay times for U.S. states from January to April 2015

We compare IFocusViz to IFocus in terms of sample complexity and run time performance. For our experiments, we use a dataset of domestic flights obtained from the website of the Bureau of Transportation Statistics [6], which provides some data about every commercial flight in the United States since 1987. We narrow our focus to data produced during the first 4 months of 2015. January, February, March, and

April had 480898, 493673, 409132, and 458311 flight records respectively.

Specifically, we are interested only in the state and the delay (in minutes) fields per record as we want to answer the following query (albeit approximately):

```
SELECT state, AVG(delay)
FROM flight_delays
GROUP BY state;
```

We set $\delta = 0.05$ (IFocus and IFocusviz will obey the visual ordering property with probability greater than 0.95), the encoding function $E(v) = \lfloor \frac{v}{10} \rfloor$, and the perceptual function $P(\cdot, \cdot) = 0$ (the most conservative perceptual function – assuming no perceptual error). Since we are grouping on the `state` field, the groups $S_1, \ldots, S_{53}$ contain the departure delay times for flights in each U.S. state.

We run our experiments on a 16GB RAM double core computer with a 2.6GHz Intel Core i5 processor. Both IFocus and IFocusviz are implemented in the Scala programming language running on Apache Spark.

Figure 3 displays the results of our experiments. Figure 3(a) shows, as expected, that for both IFocus and IFocusviz the percentage of the total data size sampled decreases as the data size increases. Furthermore, on average, IFocusviz samples less than IFocus . This is because IFocusviz converges faster since it uses the encoding function to determine what groups are still active whereas IFocus is oblivious to the encoding function. Figure 3(b) shows the run-time performance of IFocus and IFocusviz . As expected, IFocusviz has a slightly better average run-time than IFocus .

Even though we are using the most conservative perceptual function ($P(\cdot, \cdot) = 0$), IFocusviz still outperforms IFocus in terms of sampling complexity and run-time performance. The sampling strategy used is uniform sampling – for its simplicity. An alternative is stratified sampling (proportionate or disproportionate) [7]. We hypothesize that IFocusviz would have had an even better sampling complexity than IFocus if we had used a less conservative perceptual function.

Figure 2 shows the result of a single run of the IFocusviz algorithm on the flights data set. Darker states, on average, have more flight delays.

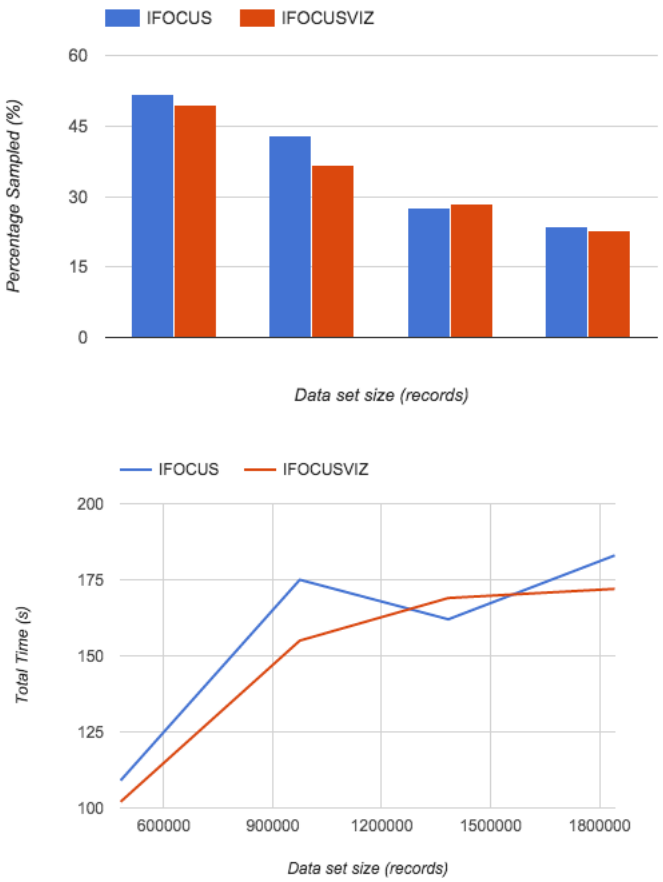

Figure 3: (a) Impact of data size on sampling percentage (b) Run-time performance of IFocus vs. IFocusviz

4 Related Work

4.1 Perceptual Functions

According to Stevens' *power law* of theoretical psychophysics [8], the perception of a value is a function of the actual value related by $p(a) = k \cdot a^{\alpha}$ where $p(a)$

and a are the magnitude of the sensation and the intensity of the stimulus respectively, k is a proportionality constant that depends on the units used, and α is the power exponent dependent on the type of stimulation [9]. For example, when the visual stimulus is brightness, S.S Stevens hypothesized that $p = 10 \cdot a^{0.33}$. Similar to Stevens' law, the Weber–Fechner law also relates the magnitude of the sensation to the intensity of the stimulus but unlike Stevens' law, the relationship between the two entities is logarithmic [10]. To obtain a perceptual function as defined in section 1, we can examine Weber's law:

$$\Delta I = K_w \cdot I$$

where I represents the initial stimulus intensity, K_w is a constant, and ΔI is the difference threshold (the *just noticeable difference* or **JND** for short). For a particular person, if we can obtain ΔI, then the perceived error for brightness, using Stevens' law, is $P = 10 \cdot (\Delta I)^{0.33}$ which is the *perceptual function* that can be used in the above algorithms. Obtaining ΔI for a particular visualization and person (let alone a group of people that will be using a particular interactive visualization) can be daunting. Because of the intractability, we can use error values obtained from previous studies [8] to approximate perceptual functions. For example, Wu & Nandi [1] suggest using a perceptual function of $P(v) = 0.02 \cdot v$ when comparing adjacent bars.

4.2 Data Visualization Management Systems

There has been some recent work [11] on integrated Data Visualization Management Systems (DVMS) based on a declarative visualization language that fully compiles the end-to-end visualization pipeline into a set of relational algebra queries. This approach is non-traditional as most visualizations today are produced in a decoupled manner: first, raw data is retrieved from a database and then using a specific visualization tool, the data is then processed and eventually rendered. With a DVMS, all database features can be made available for visualization and since all layers of the DVMS are interaction-aware, appropriate optimizations can be applied to support interactivity. Furthermore, a DVMS is equipped with database management system features relevant to interactivity. For example, with lineage query support, we can automatically link related geometric objects across views by tracking overlap in the input records that generated them.

Both IFocus and IFocusViz can be integrated into a DVMS since every layer of the DVMS is *interaction-aware* and should thus have access to (or have the ability to infer) the encoding function(s) used for interactive visualizations.

4.3 Dynamic Reduction of Query Result Sets

Sampling to reduce information presented to the user – for both efficiency and visual clarity – has been used by resolution reduction systems. ScalaR [12] is such a system that performs resolution reduction on query results. It dynamically determines if the result of a DBMS query is too large to be effectively rendered on existing screen real estate. Based on this information, ScalaR offers a three-tiered solution: insert aggregation, sampling, and/or filtering operations into the query.

4.4 Sampling Strategies

In this paper, we have left the choice of sampling strategy to the user. For our experiments, we utilized uniform sampling. Other approximate query processing algorithms use a conventional round-robin stratified sampling [13] strategy. For example, online aggregation [2] uses this sampling strategy to construct confidence intervals for estimates of averages of groups. Selected samples are not chosen a-priori. On the other hand, offline approximate query processing systems such as BlinkDB [14] and Aqua [15] select stratified samples a-priori, typically tailored to a specific workload or a small set of queries.

Uniform random sampling is strictly worse than round-robin stratified sampling when the data set is skewed. On such data sets, IFocusViz can be made to select samples using an optimum allocation stratified sampling strategy.

5 Conclusion & Challenges

In this report, we presented IFocusViz , a sampling-based approximate query processing algorithm that uses perceptual and encoding functions to converge faster towards an approximated value. Based on preliminary experiments, we find that user-specified encoding and perceptually functions can be used to display more interactive visualizations by sampling less and more quickly.

For experiments above, the most conservative perceptual function $P(\cdot, \cdot) = 0$ was used. A major challenge still remains: how can we tractably determine perceptual functions and how do perceptual functions generalize to different groups of people? As presented in this report, the use of encoding functions alone is sufficient to decrease the sampling complexity.

References

[1] Eugene Wu and Arnab Nandi. Towards Interactive Data Visualization Management Systems.

[2] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 171–182, 1997.

[3] Peter J. Haas. Hoeffding Inequalities for Join Selectivity and Online Aggregation. Technical Report 90536.

[4] Ran Canetti, Guy Even, and Oded Goldreich. Lower Bounds for Sampling Algorithms for Estimating the Average. Technical report, 1994.

[5] Eric Blais, Albert Kim, Piotr Indyk, and Sam Madden. Rapid Sampling for Visualizations with Ordering Guarantees. In *41st International Conference on Very Large Data Bases*, volume 8, pages 521–532, 2015.

[6] `http://www.transtats.bts.gov/DL_SelectFields.asp?Table_ID=236`. [Online; accessed 31-August-2015].

[7] Sl Lohr. Sampling: Design and Analysis. chapter 3, page 596. 2010. ISBN 0495105279.

[8] William Cleveland and Robert McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79:531–554, 1984.

[9] `http://www.cis.rit.edu/people/faculty/montag/vandplite/pages/chap_6/ch6p10.html`, 2002. [Online; accessed 31-August-2015].

[10] Gabriel Tzur, Andrea Berger, Roy Luria, and Michael I. Posner. Theta synchrony supports Weber-Fechner and Stevens' Laws for error processing, uniting high and low mental processes. *Psychophysiology*, 47(4):758–766, 2010. ISSN 00485772. doi: 10.1111/j.1469-8986.2010.00967.x.

[11] Eugene Wu, Leilani Battle, and Samuel R Madden. The Case for Data Visualization Management Systems [Vision Paper]. pages 903–906, 2012. doi: 10.14778/2732951.2732964.

[12] Leilani Battle, Remco Chang, and Michael Stonebraker. Dynamic reduction of query result sets for interactive visualizaton. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, pages 1–8, 2013. doi: 10.1109/BigData.2013.6691708.

[13] Stratified sampling – wikipedia, the free encyclopedia, 2015. [Online; accessed 31-August-2015].

[14] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European Conference on Computer Systems - EuroSys '13*, page 29, 2013. ISBN 1450319947. doi: 10.1145/2465351.2465355.

[15] Swarup Acharya, Phillip B Gibbons, and Viswanath Poosala. Aqua: A Fast Decision Support Systems Using Approximate Query Answers. *International Conference on Very Large Databases (VLDB)*, pages 754–757, 1999.