QueryLines: Approximate Query for Visual Browsing

Kathy Ryall, Neal Lesh, Tom Lanning, Darren Leigh, Hiroaki Miyashita and Shigeru Makino

TR2005-015 February 2005

Abstract

We introduce approximate query techniques for searching and analyzing two-dimensional data sets such as line or scatter plots. Our techniques allow users to explore a dataset by defining QueryLines: soft constraints and preferences for selecting and sorting a subset of the data. By using both preferences and soft constraints for query composition, we allow greater flexibility and expressiveness than previous visual query systems. When the user over-constrains a query, for example, a system using approximate techniques can display ïnear missesẗo enable users to quickly and continuously refine queries.

CHI 2005

QueryLines: Approximate Query for Visual Browsing

Kathy Ryall, Neal Lesh,
Tom Lanning, Darren Leigh
MERL
Mitsubishi Electric Research Labs
Cambridge, MA USA
{ryall,lesh,lanning,leigh}@merl.com

Hiroaki Miyashita, Shigeru Makino
Antennas Technology Department
Information Technology R&D Center
Mitsubishi Electric Corporation
Ofuna, Japan
{miyas, makino}@isl.melco.co.jp

ABSTRACT
We introduce approximate query techniques for searching and analyzing two-dimensional data sets such as line or scatter plots. Our techniques allow users to explore a dataset by defining QueryLines: soft constraints and preferences for selecting and sorting a subset of the data. By using both preferences and soft constraints for query composition, we allow greater flexibility and expressiveness than previous visual query systems. When the user over-constrains a query, for example, a system using approximate techniques can display "near misses" to enable users to quickly and continuously refine queries.

Categories & Subject Descriptors: H.5.2 [Information Interfaces and Presentation]: User Interfaces; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – query formulation, information filtering

Keywords: Visual query, approximate query

INTRODUCTION
Time-series data sets (e.g., finance, weather, genetics) and other data sets involving ordered sequences (e.g., census data) are especially well suited for visual query techniques because the data itself can be presented graphically. Two-dimensional data are often displayed as line or scatter plots. Exploring large datasets, however, presents a challenge. Looking at the plots sequentially is too time consuming. Looking at all the data in parallel (e.g., overlaying all of the plots in a single display) reveals little if any information; the resulting graph is often an indistinguishable blob (as shown in Figure 1(a)). Using hard constraints may over-constrain a query, resulting in few (or no) matching plots. In Figure 1(b) only four plots meet our query. However, given that visual queries are inherently approximate in nature, users may also be interested in plots that come close to meeting the constraints.

We introduce QueryLines, a flexible visual query tool that allows users to form queries consisting of soft constraints (in place of hard constraints) and preferences. As described

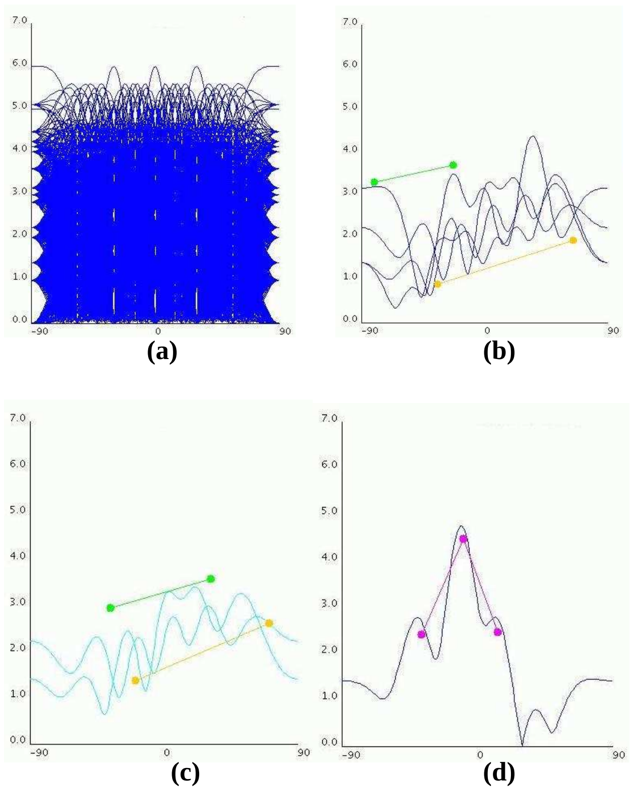

Figure 1: Example queries for a sample dataset: (a) all the plots, (b) `min` and `max` hard constraints prune the set of exact matches, (c) an over-constrained query (i.e., no exact matches) with two soft matches, and (d) a preference, `goal` query.

in the next section, soft constraints are used to divide a dataset, while preferences are used to rank those results. QueryLines introduce the notion of a "soft match" or near miss for graphs that come close to, but violate one or more of the constraints in the query. Furthermore, we allow additional flexibility for both soft constraints and preferences. As we describe below, QueryLines allow the user to search for a pattern without specifying exactly where that pattern occurs in the graph.

In the remainder of this paper we define QueryLines, and show how its combination of soft constraints and preferences supports a rich and flexible visual query language, and provide a novel interaction technique that could be added to many of today's visual query systems.

QUERYLINES

QueryLines are the graphical representation of a user's query. A user constructs a query graphically by drawing one or more query lines. A single query line represents a constraint, and has three parameters: strength, penalty and variability.

QueryLines Parameter: Strength

The ***strength*** of a query line indicates whether it is used to select or sort query results. There are two strengths:

1. **Soft constraint**: divides plots into two sets – those that fully satisfy the constraint (exact matches) and those that do not (soft matches).
2. **Preference**: further sorts the query results (i.e., plots, both exact matches and soft matches). Preferences break ties between plots that fully satisfy or equally violate the soft constraints.

Soft constraints are similar to "hard" constraints common to many visual-query systems (e.g., [2]) in that they are used to *select* the subset of plots that fully satisfy the constraints. However, rather than discard plots that violate soft constraints, we label them as "soft matches," and include them in the query results, providing users with information on near misses to their query. In Figure 1(c), for example, the two soft constraint query lines over-constrain the search; no plots meet the constraints. Two soft matches are shown; each violates the `min` query line. The soft matches are distinguished visually, by drawing them in a different color, from those that satisfy the constraints fully. Showing "near misses" will help the user adjust the constraints so that they can be satisfied.

A ***preference*** specifies an *idealized target*, and is used to *sort* the query results. Plots are ranked by the extent to which they satisfy one or more preferences. For example, in Figure 1(d), the user specifies a preference for plots with a peak near the center of the graph; the closest match is shown. Preferences provide users with a mechanism to specify desirable characteristics of a query, without overly constraining the query itself.

For each query, the user specifies which query lines are soft constraints and which are preferences. The data can then be divided into "exact matches" and "soft matches" – ***exact matches*** are those plots that *satisfy all of the soft constraints*. ***Soft matches*** *violate one or more of the soft-constraints*. Both types of matches are scored and sorted by how much they violate the preferences.

Computationally soft constraints and preferences are both represented by objective functions, but are applied differently. First, soft constraints are used to prune and sort the data set; preferences are then used to further sort the query results.

From a user's point of view soft constraints and preferences are conceptually very different. Soft constraints are the heart of the query; a user expects them to be met, or at least wants to get as close as possible to them. In contrast, preferences indicate user's priorities, in some sense "hotter and colder," and are used to rank query results (i.e., exact and soft matches) obtained from the soft constraints.

The power of the approximate query techniques comes from the ability to quickly iterate and refine the query. Through visual inspection, the user can identify which soft constraint(s) may need to be adjusted. At the same time, soft constraints can also be left in place, with soft matches providing an idea of how close the data is to the requirement. Finally, preference query lines serve as useful guides for ranking matches.

QueryLines Parameter: Penalty

For a given query line, its penalty type (as described below) is used to compute a non-negative score for each 2D plot – the larger the number the greater the violation of the query. A zero indicates that the plot satisfies the query line completely.

The extent to which a plot violates a single query line is determined by that query line's penalty and variability. A query line's ***penalty*** indicates how to compare a data point (from a particular plot) to a single query line. Any traditional constraint could be incorporated into QueryLines. We currently support four types of penalties:

1. **`minimum`**: examines a 2D plot point-wise and penalizes anything that falls below the query line.
2. **`maximum`**: examines a 2D plot point-wise and penalizes anything that falls above the query line.
3. **`goal`**: examines a 2D plot point-wise and penalizes any deviation from query line.
4. **`trend`**: examines consecutive data points and penalizes anything that does not match the direction (e.g., increasing, decreasing) of the query line.

`Min` and `max` constraints are commonly found in many query systems, and are useful for pruning data by setting absolute bounds. A `goal` query line provides users with a mechanism to look for patterns in the data. When looking for shapes it is often useful to incorporate some fuzziness into the pattern description; it is less likely that data will exactly match a given pattern specified by a user.

A `trend` query line can be thought of as a directed guideline, and is used to search for patterns of behavior over an interval of the data. Looking for a stock that generally goes up over time is one such example. `Trend` query lines are applied along the x-axis, and are evaluated based on whether the plot goes up or down between each x-tick. A plot is penalized where its direction does not match the direction of the trend.

QueryLines Parameter: Variability

The query line ***variability*** denotes along which (if any) dimension the constraint may move. There are four possibilities:

1. **`fixed`**: the query line is evaluated as specified.
2. **`x-flexible`**: the query line can move horizontally in user-defined range to find it's minimum penalty.
3. **`y-flexible`**: the query line can move vertically in user-defined range to find its minimum penalty.
4. **`both-flexible`**: the query Line can move both horizontally and vertically to find its minimum penalty.

Thus each query line is parameterized in three dimensions: strength (preference or soft constraint), penalty (`min, max, goal,` or `trend`), and variability (`fixed, x-flex, y-flex, or both-flex`). Each query is typically composed of multiple query lines. Plots are grouped and ranked based on the extent to which they satisfy a *set* of query lines.

APPROXIMATE QUERY IN PRACTICE

In this section we illustrate the approximate query techniques by walking through a series of sample queries. For illustrative purposes, for all the examples, only the top two matches are shown for each query. We begin by exploring a single `goal` query line, and the effect of changing its parameterization. The power of using both soft constraints and preferences comes from using some of each type together, as shown in subsequent examples.

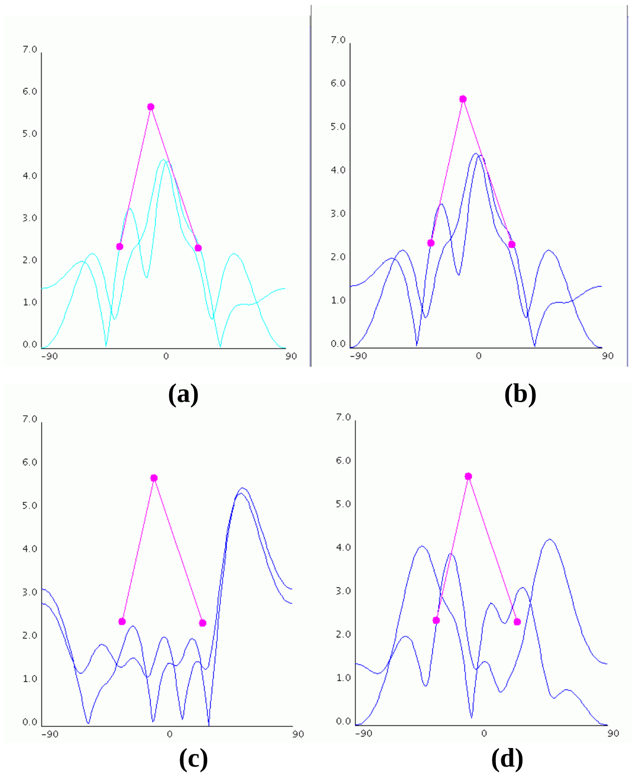

Figure 2: Parameterization of a single `goal` query line: (a) soft constraint, (b) preference, (c) preference and `x-flex`, and (d) preference and `both-flex`.

In Figure 2(a) the `goal` query line is a soft constraint. In order to meet this constraint, a plot would have to contain that pattern exactly. No such plots are in the database; the top two soft matches displayed in a light color. In Figure 2(b) the `goal` query line has been changed from a soft constraint to a preference. Since there are no soft constraints, *every* plot is an exact match. The top two plots that come closest to matching the preference are displayed, this time in a darker color because they are exact matches. Of course, these are the same two plots from 2(a).

Figure 2(c) shows the additional flexibility of adding `x-flex` to the `goal` query line; the system now returns matches that have a peak anywhere along the x-axis. The two exact matches displayed here more closely fit the shape and height of goal query line defined by the user as compared to Figures 2(a-b). Finally, Figure 2(d) shows a goal query line with `both-flex`. The peak defined by the `goal` query line can move in both the x and y dimensions.

Figure 3 shows how the user can combine different types of query lines using both soft constraints and preferences. In this example the query is composed of two query lines (a `min` and a `max`). In 3(a), the `min` query line is a soft-constraint while the `max` query line is a preference. In 3(b) the `min` query line has been changed to be a preference, while the `max` has been changed to a soft constraint. Although the positions of the query lines remain the same in both queries, the results of these two queries are quite different sets of plots. The soft constraint under constrains the problem differently in each case; the preference is used to sort the matches.

Query lines can be used and combined in a multitude of ways. For example, suppose a user is browsing through stock data. If they want to find the stock with the maximum increase in price for a given time period, they could place two preference `goal` query lines: a single-point one at a cost of zero at the beginning of the time period, and

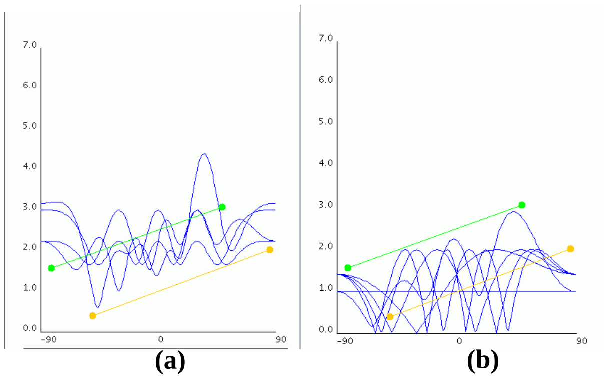

Figure 3: Comparing preference and soft constraint query lines: (a) preference `max` query line and soft constraint `min` query line, and (b) soft constraint `max` query line and preference `min` query line.

a single-point one at a maximally high price at the end of the time period. The plot that minimally violates these two preferences will have maximum gain in cost. Alternatively, the user could find the stock that went up the most number of days over a time period using a single `trend` query line.

We developed QueryLines while designing a visual browsing system for antenna data [3]; all example data sets in this paper are from the domain of antenna analysis. In our system we provide three tightly coupled graphs for the underlying antenna data: a gain plot, amplitude plot, and phase-shift plot. QueryLines can be drawn in any or all of the graphs, and are combined together to find the antenna designs which best match the total set of query lines. Traditional GUI components (e.g., pull-down menus and checkboxes) permit the user to change the strength, the penalty and the variability of each query line. A secondary window provides access to the query results including the number of exact matches found, which may be more than the number of matches requested by the user. The N matches requested by the user are separated and displayed side-by-side in two lists: "exact" and "soft" matches. Users may then select which matches to display.

DISCUSSION AND FUTURE WORK

While QueryLines provide a sketch-based interface similar to those of other systems (e.g., QuerySketch [5]), our focus is on providing users with parallel mechanisms for *specifying* a query and for *sorting* the resulting matches. The notion of a soft-match better supports users in their interactive data exploration by allowing them to see which constraints may need to be relaxed for a particular data set. The QueryLines concept could be added to enrich other sketch-based interfaces. The notion of variability in QueryLines (allowing a particular constraint or preference flexibility with respect to its positioning) is reminiscent to the notion of a "blurry match" [1]. The emphasis in that work, however, is on a shape definition language rather than on the query interaction and presentation of results. QueryLines is complimentary to this work, and could be used in conjunction with it to make an interactive system.

Our approach also has similarities to that of TimeSearcher [2]. In TimeSearcher, users draw boxes rather than the lines used in our system. A simple Timebox represents two hard constraints: the top of the box is a hard maximum constraint and the bottom of the box is a hard minimum constraint. Any plot that violates these constraints is not displayed. In our terminology, Variable Time Timeboxes extends the Timebox to allow x-flexibility. Note this is slightly more expressive than our system because the min and max lines have to move together. We have not yet explored the possibility of linking query lines in our system. However, our approach offers greater flexibility because the min/max query lines do not have to be horizontal, we offer the goal and trend QueryLines in addition to min/max constraints, and we offer y-flexibility as well as x-flexibility. More importantly, our main focus is on approximate query techniques that are not present in TimeSearcher. Our work shares a similar goal as some information visualization work providing "sensitivity" [4], the ability for a user to learn more about the data through the process of querying, and to identify which elements are most critical, or in this case have the most impact on a query. Hard constraints and their associated notion of binary satisfaction do not provide this support.

In this paper we described our approximate query techniques for searching and analyzing two-dimensional data sets. In our prototype system, our soft constraints function as hard constraints when there is data that satisfies them. QueryLines could also be used in addition to explicit hard constraints; there may be times when absolute cutoffs are required. The examples given here use line plots; our techniques would be applicable to scatter plots as well. We also note an important future direction for this work. Our prototype system relies on color-coding and a legend of sorts to provide visual feedback for QueryLines. We would like to identify and explore alternative visual feedback to enable users to quickly determine the parameters for a particular query line. Appropriately weighting the penalties when scoring plots is also an important issue, but beyond the scope of this paper.

Our approximate query techniques, soft matches supported by the *separation* of constraints and preferences, increase the flexibility and thus the power of users' queries and in turn their productivity. QueryLines, parameterized by strength, penalty, and variability create a flexible, powerful query specification language for users; combining soft constraints with preferences provides users with a powerful mechanism to separate and sort data. Displaying soft matches, plots that minimally violate soft constraints in the user's query, enable users to quickly and continuously refine their queries. Such interactive data exploration is particularly helpful when trying to understand a data set, and explore tradeoffs. The approximate query techniques presented in this paper are a novel interaction technique whose components could easily be added to many of today's visual query systems.

REFERENCES

1. R. Agrawal et. al., "Querying Shapes of Histories," Proc. Of the 21st Int'l Conference on Very Large Databases, Zurich, Switzerland, September 1995.
2. H. Hochheiser, "Interactive Graphical Querying of Time Series and Linear Sequence Data Sets," Ph.D. Dissertation, University of Maryland, May 2003.
3. D. Leigh et. al., "Exhaustive Generation and Visual Browsing for Radiation Patterns of Linear Array Antennas," Proc. of ISAP, Sendai, Japan, August 2004.
4. R. Spence, "Information Visualization," ACM Press, 2001.
5. M. Wattenberg, "Sketching a Graph to Query a Time-Series Database," CHI 2001 Extended Abstracts, Seattle, Washington, pp. 381-382, April 2001.