Intelligent Visual Analytics Queries

Ming C. Hao and Umeshwar Dayal
Hewlett Packard Laboratories, CA

Daniel A. Keim, Dominik Morent, and Joern Schneidewind
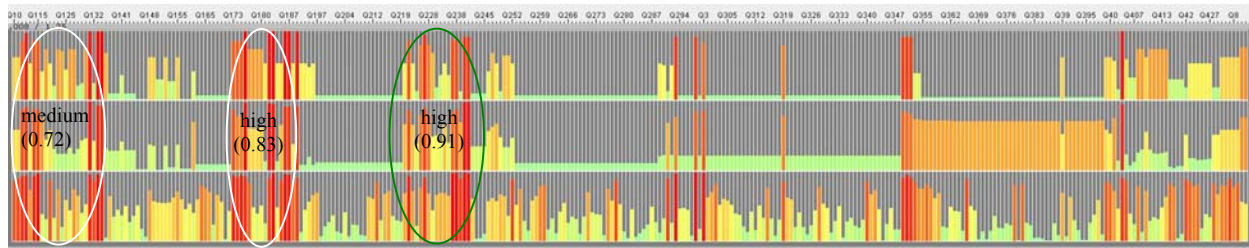University of Konstanz, Germany

Figure 1: Discover Local Correlations for Subsets of Operations in a Database Workload Application
Bars represent database operations. Color is the execution time (from green to red). The operations have low global correlation
between *execution time* and *cardinality* (0.43). But the operations in the green and white circles have high correlations (0.91-0.72)

ABSTRACT

Visualizations of large multi-dimensional data sets, occurring in scientific and commercial applications, often reveal interesting local patterns. Analysts want to identify the causes and impacts of these interesting areas, and they also want to search for similar patterns occurring elsewhere in the data set. In this paper we introduce the Intelligent Visual Analytics Query (IVQuery) concept that combines visual interaction with automated analytical methods to support analysts in discovering the special properties and relations of identified patterns. The idea of IVQuery is to interactively select focus areas in the visualization. Then, based on the characteristics of the selected areas, such as the selected data dimensions and data records, IVQuery employs analytical methods to identify the relationships to other portions of the data set. Finally, IVQuery generates visual representations for analysts to view and refine the results. IVQuery has been applied successfully to different real-world data sets, such as data warehouse performance, product sales, and sever performance analysis, and demonstrates the benefits of this technique over traditional filtering and zooming techniques. The visual analytics query technique can be used with many different types of visual representation. In this paper we show how to use IVQuery with parallel coordinates, visual maps, and scatter plots.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation - Viewing Algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques – Interaction Techniques.

Additional Keywords: Visual Analytics Query, Similarity Queries, Interactive Queries.

1   INTRODUCTION

Visualization techniques have proven to be very effective in exploring patterns in large multi-dimensional data sets, and a number of well-known visualization methods have been proposed [1].

Electronic Mail Addresses:
Ming.hao@hp.com
Umeshwar.dayal@hp.com
keim@informatik.uni-konstanz.de
Dominik.Morent@uni-konstanz.de
schneide@inf.uni-konstanz.de

Visualization often leads to immediate insight in complex phenomena, interesting patterns, or the detection of outliers [2]. In addition to overall patterns and trends, visualizations often reveal interesting local phenomena representing local patterns [3] [4]. Figure 1, for example, shows local correlations in a database operation monitoring application. Analysts are typically interested in the root-causes of the selected areas. A common way to perform the root-cause analysis is to use interaction techniques, such as filtering and zooming. However, for visual analytics in large high-dimensional data sets, these interaction methods are not sufficient, since in complex data sets a manual search becomes tedious and often impractical. New visual analytics techniques are needed to support the user in analyzing the characteristics, impacts, and relationships of selected global and local patterns.

In this paper, we propose the Intelligent Visual Analytics Query (IVQuery) that supports analysts in discovering the properties and relationships of selected patterns. IVQuery follows the Visual Analytics paradigm [5] and aims at integrating analytical and visual methods. The general idea of IVQuery is to let the analyst select an area of interest in generated visualizations (i.e., the green oval in Figure 1), identify the attributes (i.e., *execution time)* and data records that correspond to the selection, and then apply automated analysis methods to identify characteristics of the selected data, as well as their relationships to other attributes and data items. In the example in Figure 1, IVQuery helps the analyst to quickly monitor database workloads by determining which attributes (i.e., *cost, cardinality, etc.*) are the best predictors of the execution time. For the latter, state-of-the-art correlation and similarity measures are employed. Depending on the task at hand and the application domain, IVQuery allows the user to plug-in specific correlation and similarity measures, as needed.

IVQuery has been successfully applied in a number of different application scenarios, including data warehouse performance analysis, product sales, and server performance analysis, which demonstrate the benefits of this approach over traditional filtering and zooming techniques. IVQuery is a general concept and can be combined with many different types of visualization techniques. In this paper, we show how IVQuery can be integrated with parallel coordinates, scatter plots, and the recently proposed visual map framework [6].

Paper outline: The next section gives an overview of techniques for interactive pattern analysis. Section 2 describes the

related work. In Section 3, we introduce the basic concepts of Intelligent Visual Analytics Query and provide a formal problem definition and an example. Section 4 describes the IVQuery algorithm. In Section 5, we provide application examples of IVQuery. The paper closes with an outlook on future extensions.

2 RELATED WORK

In recent years, the interactive analysis and exploration of large data sets has attracted a lot of research attention. The Polaris system [7], for example, allows users to interactively generate visual representations from relational data. The user can select single dimensions or measures from the underlying relational data via drag and drop; visual representations of the data, such as line and bar charts, are constructed for these selections. The user may then refine the search, e.g., by interactively adapting the visual representations via rubber-banding or interactive queries. The *VisDB* System [8] is another well-known system that supports the exploration and analysis of large databases using pixel-based visualization techniques. The graphical user interface of the *VisDB* System allows a direct interaction with the data. Users may, for example, interactively change the query ranges, highlight corresponding pixels in different sub windows, use a projection of the visual representation to the specific color ranges, and access the attribute values that correspond to some specific pixel.

Although these techniques have been proven to be very powerful tools to reveal patterns from large data sets, they often do not provide the user enough functionality to explore the causes or impacts of those patterns. For example, if the user identifies an obvious pattern in a single dimension of a data set, it would be important to analyze the relationship to other dimensions, e.g., by identifying and visually representing the correlations to other dimensions. Such an analysis should consider global relationships (e.g., similarity between attributes), as well as local relationships (e.g., partial similarity between attributes). The XmdvTool [9] [10], for example, provides powerful techniques for dimension ordering to identify relevant relationships between single attributes. These techniques, however, often take only global measures into account, e.g., the overall correlation between attributes, but ignore the more interesting local patterns. For effective analytical reasoning, it is therefore important to support the user in determining the cause or impact of relevant patterns by integrating automated data analysis methods into the exploration process. Some approaches have been proposed in the context of time series analysis. The TimeSearcher Tool [11] [12] for example, allows analysts to interactively query and explore time-series data. Multiple time series are presented in a single aligned view, and queries can be built using time-boxes that specify local regions of interest. These time boxes are rectangular query regions drawn directly on a two-dimensional graph that shows the time series data. The idea is to find similar occurrences in the time series data, based on the selected pattern.

This paper extends these ideas to more general application scenarios by allowing more complex visual queries that intelligently integrate sophisticated analysis methods. We provide a framework to interactively select areas of interest in the visual representation of the data, which may span multiple time intervals and multiple attribute values. Based on the characteristics of the selections, the system performs complex analytics queries to identify relationships and impact factors between the selected subset of the data and the whole data set. The visual layout is then adapted to the outcome of the analysis, e.g., by representing highly related dimensions close to each other, thereby supporting the visual root-cause and impact analysis process.

3 INTELLIGENT VISUAL ANALYTICS QUERIES

3.1 BASIC IDEA

To analyze the relationships of selected global or local patterns, we propose the Intelligent Visual Analytics Query (IVQuery) framework, which is based on an iterative four step process:

1. Select a local focus area.
2. Analyze the data items, attributes, and relationships corresponding to the selected area.
3. Based on the characteristics of the selection from step 2, IVQuery automatically selects an appropriate analytic method to measure the relationship between the selected data and the overall data set.
4. Order, transform, layout and compose the results of step 3 in an appropriate visual representation.

In step 1, the user selects an area of interest in the underlying visualization, typically an area that reveals an interesting pattern. This selection may span across multiple attribute ranges. Many visualization techniques (e.g. Treemaps) also group the data according to some predefined attribute; thus the selection may also span across multiple groups. Therefore, step 2 analyzes the data items and attributes that correspond to the selected area of interest.

Based on the characteristics of the selected data, an IVQuery is performed in step 3 analyzing the relationships between the selected data and the overall data set. A number of different methods can be used to analyze relations between data items or data attributes, including data mining techniques (clustering, classification, etc.), statistical methods (correlation analysis) or similarity measures [16]. We integrated some of the most common state-of-the-art correlation and similarity measures in IVQuery. Similarity depends on the specific domain and can be defined in various ways. Additional analysis functions can be incorporated into the IVQuery framework on demand. Based on the results of step 3, the last step generates the visual layout for effective root-cause and impact analysis, by comparing attributes that are highly related to the area of interest.

Before we describe the IVQuery process in more detail, in the following we provide a motivating example that illustrates the proposed concepts. Figure 2 shows how service managers can visually analyze a *hot spot* found in performance data for data warehouse loading jobs. A typical analysis task for service managers is the identification of factors that are responsible for long loading times. Figure 2 shows the overall process of IVQuery in order to visualize the hot spot impact factors:

1. Interactively rubber-band the hot spot (5/9-5/14) with month 5, day 9-14, and attribute duration.
2. Analyze the selected rows (group *month*), columns (time interval day), and colors (attribute *duration*).
3. Measure the local correlations between *duration* and all other attributes (i.e., *delay*, *start_hr*, *rows_sec*, *rows_inserted*, etc.) based on the selected time intervals to find the root cause factors of long duration times.
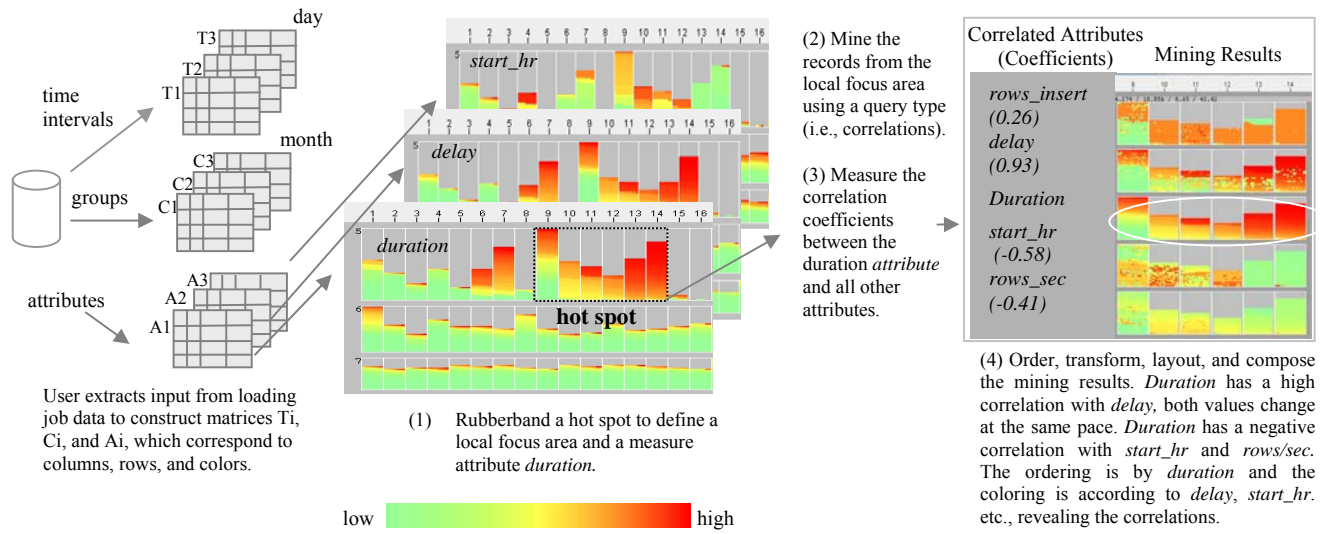
Figure 2: IVQuery Overall Process Flow on Analyzing Performance of Daily Data Warehouse Loading Jobs
group (*month*); time interval (*day*); attributes (*duration, delay,* and *start_hr*); colors (attribute values).

4. Order the attributes in Figure 2 (step 4) according to the coefficient values and place them close to the attribute *duration* in the resulting visualization. To show the correlations between attributes, IVQuery lays out all the daily jobs inside the bar. The jobs are ordered by *duration* time and the coloring is by each related attribute (i.e., *delay, start_hr*, etc.), revealing the local correlations. Usually, spotted-color in a bar indicates that the two attributes do not perfectly correlate. Service managers can find the degree of the correlations by comparing both the colors and the spotted-color areas in a bar.

From the resulting visualization in Figure 2 (step 4), the service manager can instantly display the most highly correlated attributes and find that the long *duration* time is positively correlated to the attribute *delay* (they change at the same pace; indicated by fewer spotted-colors) and negatively correlated to the attribute *start_hr* (change at the opposite pace: when the attribute *duration* is red, the attribute *start_hr* is green). These facts imply that when the *duration* is high, the *delay* is also high but the *start_hr* is low (late, it competes with many other jobs). With IVQuery this analysis can be performed interactively; the necessary calculations in the example only take about 361 milliseconds. Developing these insights by a manual search would have taken much longer, since the analyst would have had to dig through many pages of tables or charts.

With these basic concepts and functionality, we now provide a detailed definition of the Intelligent Visual Analytics Query framework. After presenting the necessary formal definitions, we show that the automated search for relationships between selected subsets of the data and the overall data set leads to a complex optimization problem in the general case; and we provide heuristics for the most common analysis tasks.

3.2   FORMAL DEFINITION OF IVQUERY

Let $D=\{D_1,...,D_k\}$ be a set of attributes (dimensions) where each dimension consists of an ordered set of n observations $D_i=\{d_i^1,...,d_i^n\}$. The order of the observations can be based on the values of a selected ordinal or continuous attribute, or the time stamp of an attribute when dealing with time related data. The dimensions may belong to a number of groups (see Figure 3).

An Intelligent Visual Analytics Query allows the selection of a subset $D' \subseteq D$, where $D'=\{D'_{i_1},...,D'_{i_l}\}$ consists of il dimensions, $ij \in \{1,.., k\}$ and h data values $D'_i=\{d_i^z,...,d_i^{z+h}\}$. The selection is done interactively by the analyst by selecting an area of interest in the underlying visualization $V(D)$ of D. It may contain multiple data objects across multiple intervals. The goal of an IVQuery is to analyze the data for subsets that are related to the selected data. More formally, the goal is to effectively retrieve a target set $D'' \subseteq D$ with:

$$Rel(D',D'') \to max$$

where $Rel(X,Y) \to rel$ defines a relevance function that assigns a relevance value $rel \in R,\ 0 \leq rel \leq 1$ to each pair of sets (X,Y) with $X, Y \subseteq D$. The higher the relevance value is, the stronger the relationship between the D' and D''. Based on these definitions two problems arise: The first one is the question of how to determine D'' from the data set D and the second one is how to compute $Rel(D',D'')$ for arbitrary D', D''. Since the input data set D can be seen as a matrix with columns D_i and rows $d_{i,}^j$, an arbitrary selection of a subset of the data D' results in a *(h*l)* matrix by selecting multiple data values h across multiple dimensions l. If the user selects only h values from a single dimension or a single record across l dimensions, D' would result in an l- (resp. h-) dimensional vector. In general, there are

$$\binom{k*n}{h*l}$$

possible selections of subsets $D'' \subseteq D$ of size $h*l$ that may be considered, where l is the number of dimensions and h is the number of rows of the selected data matrix D'. An automatic analysis of all subsets is computationally infeasible due to the exponential growth of the binominal coefficient. For a specific visualization technique, the number of possible selections is lower, since the visualization restricts the selection to specific dimensions and groups of data items.

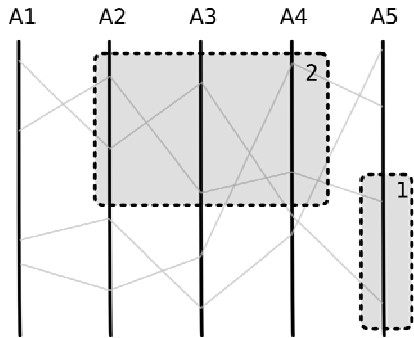Figure 3: Scheme for IVQuery selection on ordered data (based on VisMap)

Figure 4: Scheme for IVQuery selection on unordered data (Parallel Coordinates Example)

Figure 5: Scheme for IVQuery selection on unordered data (Scatter Plot Example). The group membership is mapped to color.

For a given selection, there are multiple possible analyses to determine the target data set *D''* with maximal relevance. Our IVQuery algorithm provides effective heuristics to focus on meaningful use cases when looking for relationships in the data. To determine *Rel(D',D'')* for arbitrary *D', D''*, IVQuery incorporates a number of analysis, similarity and correlation functions. Depending on the characteristics of the selected data *D'*, the analysis functions must be applied appropriately. If *D'* contains for example a single attribute, the pair-wise correlation to all other attributes is computed. If *D'* contains a set of attributes the similarity to other sets of attributes *D''* is computed by pair-wise similarity measures

$$SIM(D_i, D_j) \quad \text{with } D_i \in D' \text{ and } D_j \in D''.$$

The next subsection describes a number of common meaningful use cases supported by IVQuery. The use cases depend on the structure of the selected data *D'*.

3.3 TYPES OF IVQUERY

As shown in the previous section, performing an exhaustive search for relationships in the data leads to a complex optimization problem in the general case. However, many common analysis tasks, such as global correlation analysis or global similarity measures, do not require the comparison of all possible subsets of the data. In many cases, it is sufficient to pair-wise compare single dimensions of the data (e.g., in global correlation analysis) or to analyze aligned subsets of the data across a fixed number of dimensions (e.g., to analyze partial correlations). The goal of our IVQuery algorithm is to support a number of common analysis tasks. Based on the characteristics of the interactively selected data, we have identified five common analytics scenarios, which correspond to general visual analytics tasks and are independent of the underlying visualization
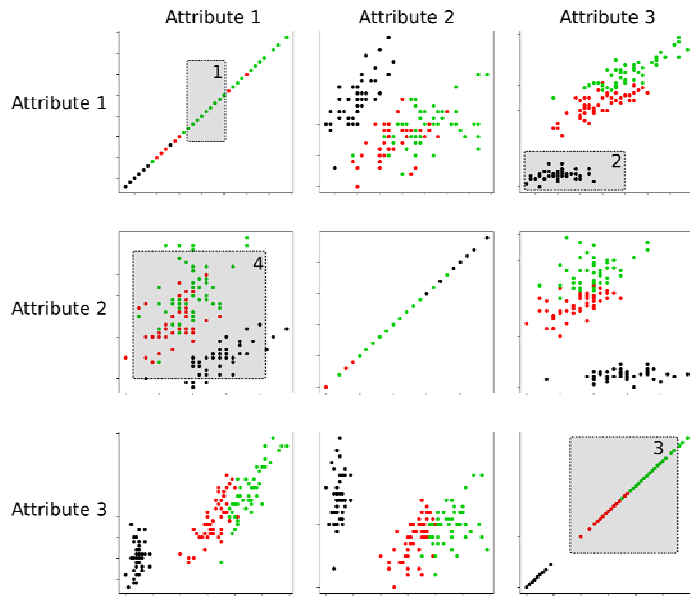
technique. We show how the IVQuery concept can be combined with different visualization techniques and different types of data, e.g. ordered (Figure 3) and unordered data (Figures 4 and 5). According to the characteristics of the interactively selected data, IVQuery automatically identifies the analysis task and selects one of our standard cases and the appropriate analysis functions. The next subsection describes these standard cases in detail. Note that the cases represent only a reasonable default behavior: The user can also select the desired analysis type manually.

CASE 1: ONE ATTRIBUTE IN ONE GROUP

The most common case when analyzing relations in the data is the selection of one continuous interval across a single attribute within one group. This corresponds to selection 1 in Figures 3, 4, and 5. Note that the selection of the whole bar is just a special case with an interval that spans all attribute values. In this case, there is no distinction between ordered and unordered data. In data sets without grouping, IVQuery searches for other attributes that are most closely related to the selected subset. In the case of grouping, the algorithm only searches for relevant attributes within the selected group. The results can be visualized by displaying all attributes that are relevant for the selection or by ordering the attributes within the group according to their relevance.

Note that an ordering of attributes according to their relevance with respect to a selected attribute, which is supported by a number of existing tools, is a special instance of this case.

CASE 2: MULTIPLE ATTRIBUTES IN ONE GROUP

Typical examples for a selection over multiple attributes within one group are shown in selection 2 in Figures 3, 4, and 5. Figure 2 shows there is a high correlation between attributes duration and delay.

If multiple groups are involved, we have to distinguish between visualizations that have a designated data ordering attribute (Figure 3) and visualizations without consistent data ordering (Figures 4, 5). While visual maps order all items in each bar from left to right, typically according to a time stamp or some other attribute value, the items in the parallel coordinates plot are sorted differently on each axis, based on the value of the corresponding attribute. This must be considered in IVQuery when selecting an analysis query type.

For ordered visualizations that have a grouping attribute, as in Figure 3, the default of our IVQuery algorithm analyzes the other groups to identify groups that are relevant on the selected interval by considering only the selected attributes. The result can than be used for a reordering of the groups, to place groups close to each other that are similar in the selected area. It is also possible to analyze the selected group and identify other attributes that are relevant to the selected attributes.

For unordered visualizations without grouping, such as the parallel coordinates plot in Figure 4, other methods are necessary. The default option of IVQuery is to perform a clustering to find relevant attributes. Other approaches are possible depending on the specific visualization. After the selection of area 2 in the scatter plot matrix in Figure 5, for example, IVQuery analyzes the scatter plot matrix to search for other scatter plots (with different attribute pairs) in which the selected items show similar patterns.

CASE 3: ONE ATTRIBUTE IN MULTIPLE GROUPS
The third case to select one attribute in multiple groups is shown in selection 3 in Figure 5. IVQuery handles it by applying case 1 independently for each selected group, which means that IVQuery searches within each group for other attributes that are closely related to the selected subset. The results are individual sets of relevant attributes for each selected group.

CASE 4: MULTIPLE ATTRIBUTES IN MULTIPLE GROUPS
The fourth case defines selections that span multiple attributes over multiple groups as illustrated by selection 4 in Figure 3 and Figure 5. In this case, for ordered data IVQuery searches for other intervals that are related to the selected attributes and groups; and for unordered data, it performs a clustering and then searches for related data points.

CASE 5: MULTIPLE ATTRIBUTES IN MULTIPLE GROUPS WITH MULTIPLE DISJOINT INTERVALS
This is the most general case and corresponds to multiple selections within one one query that range across multiple attributes, multiple groups, and multiple intervals. An example would be the combination of selections 2 and 4 in Figure 3. Even for this very general case, a relevance calculation is possible by combining the single relevance results of each selection. This can be done in various ways. One option is to let the user specify the importance of each selection by assigning weights and to calculate a weighted normalized average. Crucial for this to be useful is that all selections allow the application of at least one common relevance measure to compute the relevance among multiple selections. If we have, for example, one selection of type 4 in figure 3, only a relevance computation over the ordering attribute is possible.

4 IVQUERY ALGORITHM

4.1 ALGORITHM

With a set of attributes D, a selected subset D' of those attributes, and an optional ordering/grouping attribute (D_o/D_g) as input, the algorithm analyzes the properties of D'. Based on these properties the algorithm determines the best approach for the relevance

```
Procedure: Intelligent Visual Analytics Query
Input: Set of attribute dimensions D, interactive selection D' ⊆ D,
       Ordering attribute D_o ∈ D, Grouping attribute D_g ∈ D
Output: D ordered by relevance

if D_g = ∅ or |D'_g| = 1 then                    /* one group selected */
   if |Attribs(D')| = 1 then            /* one attribute selected */
      // Case 1:  (One Attribute in One Group)
      /* consider all attributes of the selected group          */
      foreach attrib in Attribs(Groups (D')) do
         /* only observations in the selected group and interval are
            considered                                          */
         d = filter (attrib, Groups(D'), Interval(D'));
         rel [attrib] = calculateRelevance(D', d);
   else                                  /* multiple attributes selected */
      // Case 2:  Multiple Attributes in One Group
      foreach group in Groups(D) do
         /* only selected attributes are considered in each group   */
         foreach attrib in Attribs(D') do
            d' = filter(attrib, Groups(D'), Interval(D'));
            d = filter (attrib, group, Interval(D'));
            r = calculateRelevance(d', d);
            rel [group] = combine(rel [group], r);
else                                     /* multiple groups selected */
   if |Attribs(D')| = 1 then               /* one attribute selected */
      // Case 3:  One Attribute in Multiple Groups
      /* handle as case 1 for each involved group              */
      foreach group in Groups(D') do
         ... (see case 1);
   else if D_o ≠ ∅ then    /* ordered and multiple attributes selected */
      // Case 4:  Multiple Attributes in Multiple Groups
      /* search over ordering attribute                        */
      foreach interval in D_o with |Interval| = |D'| do
         foreach group in Groups(D') do
            foreach attrib in Attribs(D') do
               d' = filter (attrib, group, Interval(D'));
               d = filter (attrib, group, interval);
               r = calculateRelevance(d', d);
               rel [interval] = combine(rel [interval],r)
   else                     /* unordered and multiple attributes selected */
      /* do not allow this case or provide a visualization dependent
         approach                                              */

return D, ordered by descending rel ;
```

Table1: IVQuery Algorithm

calculation and returns the whole data set D ordered by relevance for the given input D'. The IVQuery algorithm is shown in Table 1.

4.2 RELEVANCE MEASURES

Based on the characteristics of D' and D'', IVQuery automatically selects a default technique to compute $Rel(D',D'')$. The choice depends, for example, on the data types comprehended in D' and D''. Note that these functions represent only a reasonable default behavior and may be adapted by the user, depending on the task at hand. For correlation analysis we included, among others, the Pearson Correlation r into the IVQuery framework:

$$ r = \frac{\sum_{i=1}^{n} (A_{1i} - \bar{A}_1)(A_{2i} - \bar{A}_2)}{\sqrt{\sum_{i=1}^{n} (A_{1i} - \bar{A}_1)^2 \sum_{i=1}^{n} (A_{2i} - \bar{A}_2)^2}} $$

This function computes the pair-wise correlation between bi-variate data A_{1i} and A_{2i}. If two dimensions are perfectly correlated, the correlation coefficient is 1, in case of an inverse correlation -1. To determine the similarity between dimensions, IVQuery employs similarity measures such as the normalized Euclidean distance:

$$ Sim(A_i, A_j) = \sqrt{\sum_{i=0}^{N-1} (b_i^1 - b_i^2)^2} $$

where

$$ b_i^j = \frac{a_i^j - \min(A_j)}{\max(A_j) - \min(A_j)} $$

Query Results

(A) Monthly Regional Sales Visual Map
Select US and France in Month 12 *to show the reasons for their* high product sales

Layout: group(*Region*); time interval (*month*); attributes
(*$Amount, Quantity, #of Visits*...etc); colors (attribute values)

(B) US Product sales *$Amount* correlates more to *Quantity* than #of Visits
- high *Quantity* values are placed in high *$Amount* areas (top of the bar, inside the black oval) colored red and burgundy.
- much fewer spotted-color pixels

(C) France Product Sales *$Amount* correlates more to the *#of Visits*
- high #of *Visits* values are in the high *$Amount* areas (top of the bar, inside the black oval) and colored red, burgundy, and blue.
- much fewer spotted-color pixels appeared in the #of Visits graph than in the *Quantity* graph.
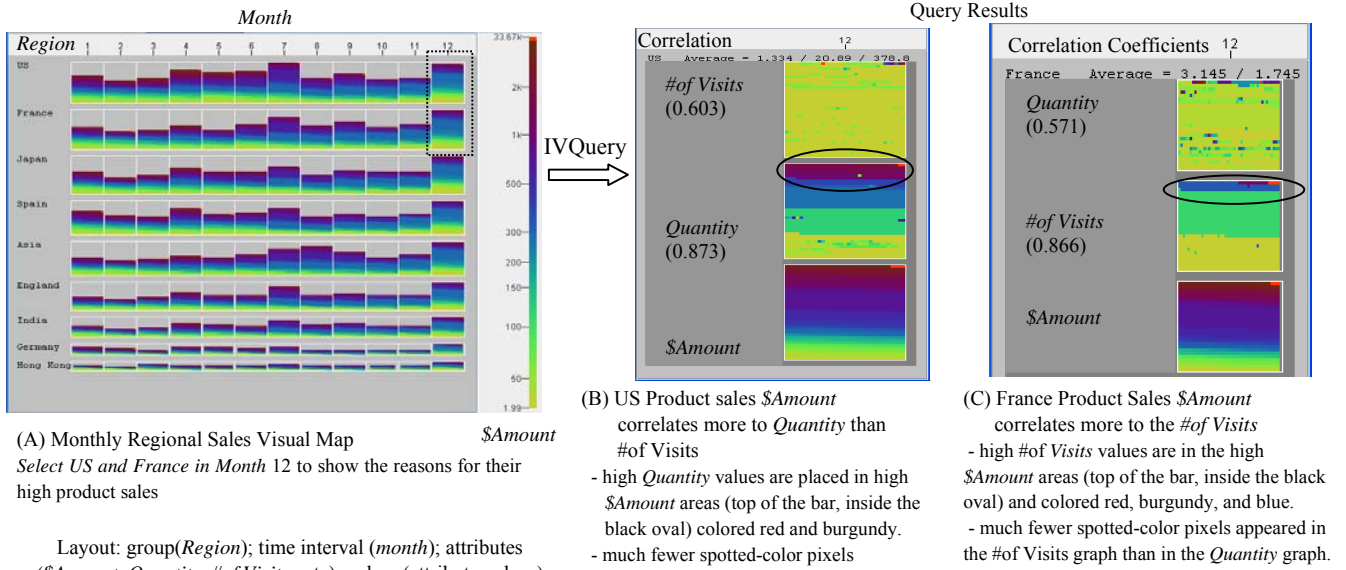
Figure 6: IVQuery Discovers Different Reasons for US and French Sales in *Month* 12

Additionally, we use techniques for partial similarity analysis, such as the synchronized similarity [16]:

$$Sim_{Syn}(A_k, A_l) = \max_{i,j}\left\{(j-i) \mid (0 \le i < j < N) \wedge \sqrt{\sum_{z=i}^{j} c_z} < \varepsilon\right\}$$

where

$$c_z = \left(b_z^k - b_z^l\right)^2$$

Depending on the application scenario the unsynchronized partial similarity may also be used [16].

To identify groups of attributes and data items that are relevant cluster and classification techniques may also be appropriate. We integrated the well-known *k*-means clustering approach and a nearest neighbor classification algorithm [13] to perform these operations. Note that there are a large number of sophisticated data analysis (clustering, classification, etc.), similarity and correlation methods proposed, that can be included into the IVQuery framework as needed.

4.3 REARRANGEMENT OF THE VISUAL LAYOUT

Based on the result of the analysis step, the visual layout determined in step 4 is rearranged according to the relationships between the selected area and the determined subsets of the data with maximal relevance. The goal is to adapt the layout so that related data subsets are located close to each other. For a given similarity vector S, the optimal arrangement of dimensions is given by a neighborhood relation *N* such that:

$$\sum N_{Query, Subset} * S\left(D_{Query}, D_{Subset}\right) \to \min$$

The neighborhood relation is one if the *Query* and the *Subset* are neighboring in the visualization, otherwise zero. If we minimize the above formula, we obtain an optimal visual representation which retains relationship between *Query* and *Subset*. To instantiate the optimization criterion, we employ a number of state-of-the art similarity and correlation measures and apply them to the visualization techniques used. Note that both visualization techniques, parallel coordinates and visual maps, are based on a linear one-dimensional arrangement. Other visualization types

may require different ways for expressing relationships. The importance of different subsets for the selected area can also be expressed in other ways, for example, by their size or color.

5 APPLICATION EXAMPLES

5.1 VISUAL MAPS EXAMPLES

To visualize correlations and similarities among multiple attributes, we use visual map, a spreadsheet-like row and column layout [6]. Rows represent groups of dimensions (e.g., *region* and *country*), and columns represent data intervals (e.g., time interval, *day*, *month*, and *quarter of a year*). The color is the attribute value of a data item, e.g., *duration, $Amount*, and *CPU Utilization*.

USING CORRELATION MEASURES TO DISCOVER REASONS FOR HIGH PRODUCT SALES

To face today's business challenges, sales analysts want to correlate customer purchase behavior with product selling and promotion. They want to know the reasons (selling attributes) for high product sales.

Figure 6 (A) is a yearly sales analysis visual map. Each sales region (bar) is filled with the number of invoices (pixels) for the corresponding month. The colors represent invoice sales *$Amount*. *US* and *France* in *month* 12 have the most invoices and the highest *$Amount* (above 1K, red and burgundy). To find the reasons for this, the analyst selects one attribute (*$Amount*) in two groups (i.e., *US* and *France*) performing the IVQuery (case 3). IVQuery mines the selected invoices and computes the corresponding correlation coefficients between the attribute *$Amount* and all other sales attributes (e.g., *Quantity, #of Visits, locations,* etc.).

Figures 6 (B) and 6 (C) show the mining results from IVQuery. The invoice ordering in each bar is by *$Amount* and the coloring is according to the most relevant attributes (*Quantity, #of Visits ...*) to show the correlations. Spotted-color pixels indicate that the two attributes do not perfectly correlate. In Figure 6 (B), the *$Amount* for US product sales correlates more to the customers buying more items (*Quantity*, 0.873) than to the number of times the customers come back (#of Visits, 0.603), because the invoices

in the black circle in Figure 6 (B) show that the high *$Amount* invoices (at the top of the bar) have the high *Quantity* values (red and burgundy). This indicates that the values of both attributes *$Amount* and *Quantity* change at the same pace. *$Amount* correlates less to the number of times the customers come back (*#of Visits)*. There are many one time customers (yellow) that appear in the high *$Amount* areas (the top of the bar). However, in Figure 6 (C), the highest sales in France are because their customers come back more frequently (more red, burgundy, and blue; 0.866). The correlations with *Quantity* have many spotted-color pixels in the graph. After knowing their customer purchase behaviors, sales analysts are able to plan their new sales strategies.

USING SIMILARITY MEASURES TO FIND SIMILAR SERVER PERFORMANCE BEHAVIOR

To manage a server workload effectively, analysts need to know how to distribute the jobs among the servers to avoid overloading the system. To do this, the analyst needs to know which servers are similar to which other servers and have similar performance behaviors, such as the percent of *CPU utilization*, *Network Bytes* transferred, *queue length*, and *disk usage* over time.

Figure 7A shows analysts of daily server performance. Rows (groups) contain server names. Columns contain time intervals (*year-month-day*). Each time interval contains a number of 5-minute measurements. The color of each measurement represents the value of the corresponding attributes.

To classify servers with similar performance behaviors in a time interval, one needs to perform IVQuery (case 2), multiple attributes in one group. The analyst selects *Server 2*, attributes (*CPU Utilization* and *Network Bytes*) and time interval (*9/18-9/28*) to compare the *Server 2* performance with that of other servers and presents the results in Figure 7B. Servers with similar performance relationships are arranged from top to bottom in Figure 7B according to the similarity measures. The results are returned immediately as the calculations only take a few hundred milliseconds (613ms on average for this example).
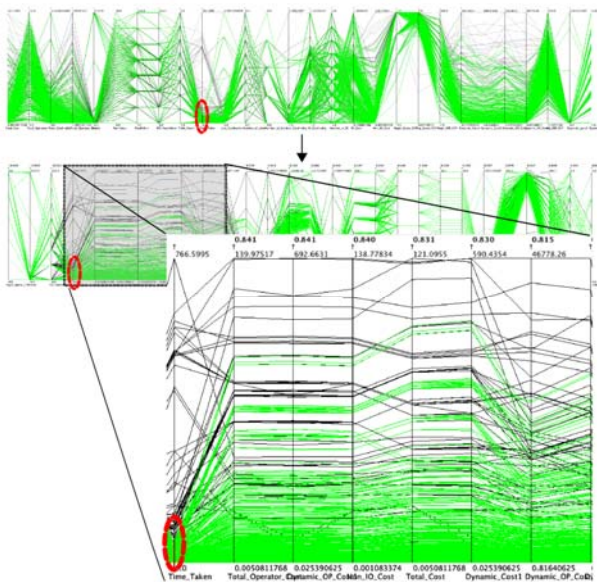

Figure 8: Discover partial correlations in parallel coordinates. The axes are reordered in a way that attributes with a high correlation on the highlighted (green) subset are placed close to the selected attributes.
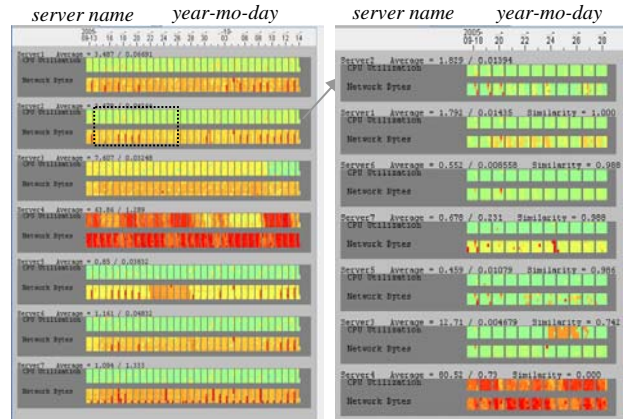

Figure 7A: Starting an IVQuery (use IVQuery case 2) select: group (*Server 2*); attributes (*CPU Utilization* and *Network Bytes*)

Figure 7B: After IVQuery discover *Servers* 1, 6, 7, and 5 have very similar performance patterns to that of *Server 2* from 9/18 -9/28

From the top of the list in Figure 7B, analysts can instantly identify the servers that are most similar to *Server 2*: *Servers 1, 6, 7,* and *5* (over 0.986). *Server 4* is very different from the other servers (at bottom of list). The analyst needs to find the bottleneck of this server. Using IVQuery, analysts are able to balance the workload among servers and enhance the system throughput.

5.2   PARALLEL COORDINATES EXAMPLE

A well-known technique for the visualization of multi-dimensional data is the parallel coordinate technique [14] [15]. IVQuery allows the selection of interesting subsets of the data, for example, by selecting an interval of one attribute. After calculating the pair-wise correlation between the selected attribute and all other attributes based on the selected subset, the axes are reordered in such a way that highly correlated attributes are placed close to the selected attributes. Positively correlated attributes are placed to the right of the selected attribute in descending order, negatively correlated attributes to the left in ascending order. This allows the analyst to discover dependencies between the attributes based on the selected local visual space, as the order of the axes is crucial for the expressiveness of the visualization [16].

Figure 8 shows an example query (case 1) on a database workload data set based on the modified parvis tool [17]. After the selection *(*red circle) of some data items with low values on the attribute *Time taken,* the axes are reordered with their correlation values to *Time taken* displayed above the axes in bold face. The enlarged image shows the details of the positively correlated attributes. The positive correlation manifests itself by mostly parallel lines with almost no intersections. The reordering by an IVQuery allows one to spot relationships of local subsets between the attributes that were not visible before. Note that for other selections different attribute orderings are returned.

5.3   SCATTER PLOTS EXAMPLE

Multi-dimensional data sets can also be visualized by using a scatter plot matrix. If a pair-wise comparison of all dimensions is required, the number of scatter plots soon gets too big for manual analysis ($O(n^2)$ with n attributes). The IVQuery supports the analyst by allowing a selection of interesting local patterns and automatically returning a filtered and rearranged scatter plot matrix showing only relevant plots. Figure 9 shows a selection (case 2) of a local pattern on the Detroit Data Set [18] that has a high correlation. The selected items are highlighted in the other
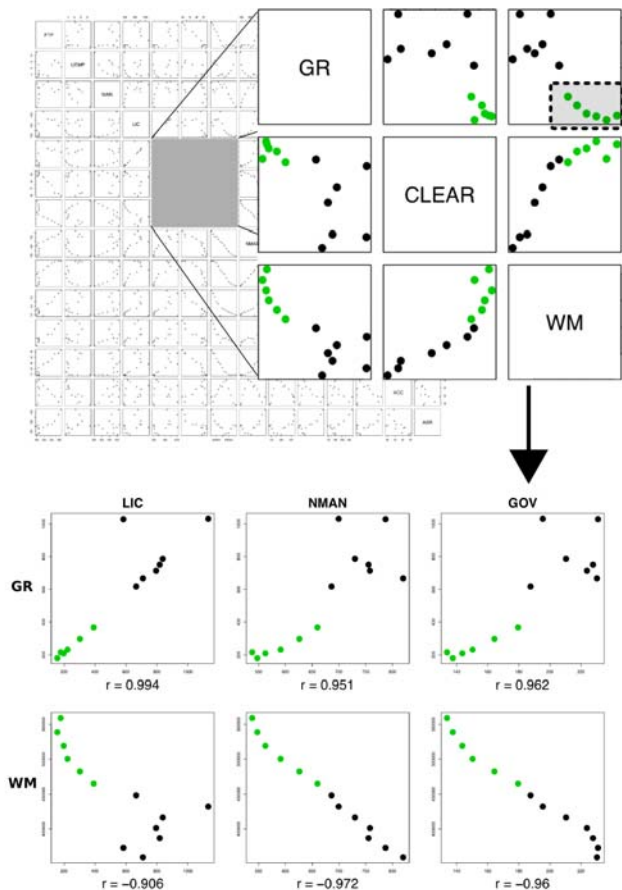
Figure 9: An IVQuery for the most similar scatter plots on the selected subsets discovers that the attributes *LIC, NMAN* and *GOV* have the highest absolute correlations to *GR* and *WM*.

scatter plots, and the IVQuery searches for other attributes that are highly correlated with both of the two selected attributes (GR and WM). The result on the bottom of Figure 9 shows that the attributes LIC, NMAN and GOV have the highest local correlations on the selected subset.

6 CONCLUSION AND FUTURE WORK

In this paper, we present Intelligent Visual Analytics Query, a new concept that provides methods to intelligently combine interactive visualizations with state-of-the-art analytics techniques. Depending on the selected area and the properties of the data set, IVQuery algorithm performs different types of analyses and returns the most relevant data subset to the analyst. In this paper, we describe five different types of analyses, which are based on different correlation and similarity measurements. The IVQuery framework can easily be extended by additional correlation and similarity metrics. The results of the analyses are presented in an interactive visualization to reveal the relations across different attributes. We apply IVQuery to mine data relationships and root-causes of enterprise data warehouse, sales, and server performance problems. From the recent feedback of system administrators, they are able to use IVQuery to monitor the progress of running jobs and to manage their server workload in real-time. Future research will focus on the automated selection of the best suited visualization for a given IVQuery.

REFERENCES

[1] D. A. Keim. Information Visualization and Visual Data Mining, IEEE Transactions on Visualization and Computer Graphics (TVCG) 1, (8) pages 1-8, 2002.

[2] R. Vliegen, J. J. van Wijk, E.-J. van der Linden. Visualizing Business data with Generalized Treemaps, IEEE Symposium on Information Visualization 2006.

[3] S. Card, R. Rao: *Exploring Large Tables with the Table Lens*. Proceedings of the ACM Conference on Human Factors in Computing Systems, 1995.

[4] Furnas: *Generalized Fisheye Views*. Proceedings of the ACM Conference on Human Factors in Computing Systems, 1986.

[5] J. Thomas and K.A. Cook. Illuminating the path: Research and Development agenda for Visual Analytics. IEEE Press, October 2005.

[6] M. Hao, U. Dayal, D. Keim, T. Schreck. A Visual Analysis of Multi-Attribute Data Using Pixel Matrix Displays. In Proc. VDA07, 2007.

[7] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. IEEE Transactions on Visualization and Computer Graphics, 8(1):52–65, 2002.

[8] D. A. Keim and H.-P. Kriegel. VISDB: Database exploration using multidimensional visualization. IEEE Computer Graphics and Applications, 14(5):40–49, 1994.

[9] M. O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. In Proc. Visualization 94, Washington, D.C.

[10] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In VISSYM '03: Proceedings of the Symposium on Data Visualisation 2003, pages 19–28, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[11] B, P. Aris, C. Plaisant, A. Khella, B. Shneiderman. Interactive Pattern Search in Time Series, Proceedings of Conference on Visualization and Data Analysis, VDA 2005, SPIE, Washington DC (2005): 175-186.

[12] H. Hochheiser, B. Shneiderman, Dynamic Query Tools for Time Series Data Sets, Timebox Widgets for Interactive Exploration, *Information Visualization 3*, 1 (March 2004), 1-18.

[13] J. Han, M. Kamber: *Data Mining: Concepts and Techniques, 2nd Edt.* Morgan Kaufmann Publishers, 2006.

[14] A. Inselberg and B. Dimsdale. Parallel coordinates: a tool for visualizing multi-dimensional geometry. In VIS '90: Proceedings of the 1st conference on Visualization '90, pages 361–378, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[15] A. Inselberg. The plane with parallel coordinates. The Visual Computer, V1 (4):69–91, 1985.

[16] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In INFOVIS '98: Proceedings of the 1998 IEEE Symposium on Information Visualization, page 52. IEEE, 1998.

[17] Ledermann: Parvis – Tool for parallel coordinates visualization. http://home.subnet.at/flo/mv/parvis/index.html

[18] Gunst, Mason: Data on annual homicides in Detroit, 1961-73, from Gunst & Mason's book 'Regression Analysis and its Application', Marcel Dekker. Contains data on 14 relevant variables collected by J.C. Fisher. Source: http://lib.stat.cmu.edu/datasets/detroit