Visual Hierarchical Dimension Reduction

by

Jing Yang

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

December 2001

APPROVED:

Professor Matthew O. Ward, Thesis Advisor

Professor Elke A. Rundensteiner, Thesis Co-advisor

Professor Micha Hofri, Head of Department

Abstract

Traditional visualization techniques for multidimensional data sets, such as parallel coordinates, star glyphs, and scatterplot matrices, do not scale well to high dimensional data sets. A common approach to solve this problem is dimensionality reduction. Existing dimensionality reduction techniques, such as Principal Component Analysis, Multidimensional Scaling, and Self Organizing Maps, have serious drawbacks in that the generated low dimensional subspace has no intuitive meaning to users. In addition, little user interaction is allowed in those highly automatic processes.

In this thesis, we propose a new methodology to dimensionality reduction that combines automation and user interaction for the generation of meaningful subspaces, called the visual hierarchical dimension reduction (VHDR) framework. Firstly, VHDR groups all dimensions of a data set into a dimension hierarchy. This hierarchy is then visualized using a radial space-filling hierarchy visualization tool called Sunburst. Thus users are allowed to interactively explore and modify the dimension hierarchy, and select clusters at different levels of detail for the data display. VHDR then assigns a representative dimension to each dimension cluster selected by the users. Finally, VHDR maps the high-dimensional data set into the subspace composed of these representative dimensions and displays the projected subspace. To accomplish the latter, we have designed several extensions to existing popular multidimensional display techniques, such as parallel coordinates, star glyphs, and scatterplot matrices. These displays have been enhanced to express semantics of the selected subspace, such as the context of the dimensions and dissimilarity among the individual dimensions in a cluster. We have implemented all these

features and incorporated them into the XmdvTool software package, which will be released as XmdvTool Version 6.0. Lastly, we developed two case studies to show how we apply VHDR to visualize and interactively explore a high dimensional data set.

Acknowledgements

I would like to express my greatest gratitude to my advisors, Matt and Elke, for their patient guidance and invaluable contributions to this work. I often think that I am very lucky that I can have two such excellent professors as my advisors at the same time. I would like to thank Prof. David Brown for being the reader of this thesis and giving me much valuable feedback.

I would like to thank my team members, Geraldine Rosario, who knows a lot about statistics and helped me on that, Punit Doshi, who cooperates with me well in developing XmdvTool, Brandon Light, who maintains XmdvTool web pages for us, and Ying-Huey Fua, who handed over very clean and clear codes to me thus making my implementation much easier.

I also want to thank Andreas Koeller, since the latex thesis template provided by him made my thesis writing process much easier, and Hong Su and Xin Zhang, who helped me a lot on learning latex.

A special thank is to my husband Zheng Zuo, who gave me many useful suggestions for this work and much more.

This work is funded by NSF under grants IIS-0119276 and IIS-9732897.

Contents

List of Figures

Chapter 1

Introduction

1.1 Multi-Dimensional Visualization Motivation

Because of the rate of technological progress, the amount of data that is stored in computers is increasing rapidly [1]. Researchers from the University of Berkeley estimate that every year about 10^{15} bytes of data is generated, with 99.997% only available in digital form [1]. Most data is collected because people believe that it is a potential resource of valuable information. However, finding the valuable information hidden in the data is a hard task. One important approach to supporting the human in analyzing and exploring such large amounts of data is to graphically present the data to the human and then to allow the human to apply his or her perceptual abilities to make sense of the data, namely "data visualization".

Multivariate visualization is one sub-field of data visualization that focuses on Multi-dimensional data set. Multidimensional data set can be defined as a set of data items D, where the i^{th} data item d_i consists of a vector with n variables, (x_{i1}, x_{i2}, ..., x_{in}). Each variable may be independent of or interdependent with one or more of the other variables. Variables may be discrete or continuous in nature, or take on symbolic (nominal) values.

Many multivariate visualization techniques and systems have emerged during the last three decades, such as glyph techniques [2, 3, 4, 5], parallel coordinates [6, 7], scatterplot matrices [8], pixel-level visualization [9], and dimensional stacking [10]. Each method has strengths and weaknesses in terms of the data characteristics and analysis tasks for which it is best suited.

Recently, lots of effort has been focussed on high dimensional data set visualization. In the next section, we will discuss open challenges in visualizing high dimensional data set.

1.2 Open Challenges in Visualizing High-Dimensional Data Sets

With the development of database management systems and the information technology industry, high dimensional data sets have been generated in many areas such as data warehousing, multimedia, document visualization, census and so on. High dimensional data sets can have hundreds or even thousands of dimensions. The need for visualizing high dimensional data sets to facilitate analysis and exploration has steadily increased.

Traditional visualization techniques for multidimensional data sets, such as parallel coordinates, glyphs, scatterplot matrices, and dimensional stacking, do not scale well with high dimensional data sets. For example, Figure 1.1 shows the Iris data set, which contains 4 dimensions and 256 data items, in parallel coordinates. Individual data items can be seen clearly from the display. Figure 1.2 shows a subset of the Census Income data set, which contains 42 dimensions and 200 data items in Parallel Coordinates. The number of data items in this display is less than that of Figure 1.1, however, individual data items cannot be seen clearly from this display since the number of dimensions has greatly increased. A large number of axes crowds the figure, preventing users from detecting any details. Moreover, the figure is so complex that it requires a long time to refresh the

display. Thus the response time when users interactively play with the data set is unacceptable. The problems of the traditional visualization techniques for high dimensional data sets can therefore be summarized as “cluttered” and “slow”.

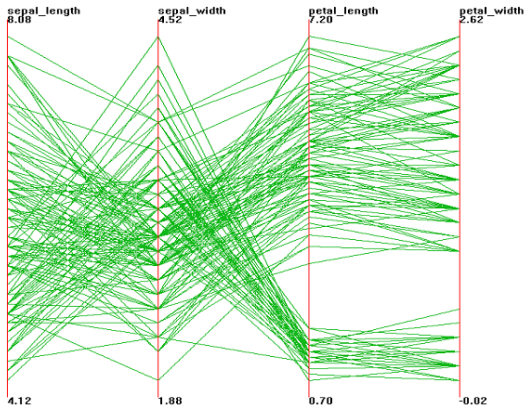

Figure 1.1: Iris data set (4 dimensions, 256 data items) in Parallel Coordinates. Individual data items can be seen clearly.

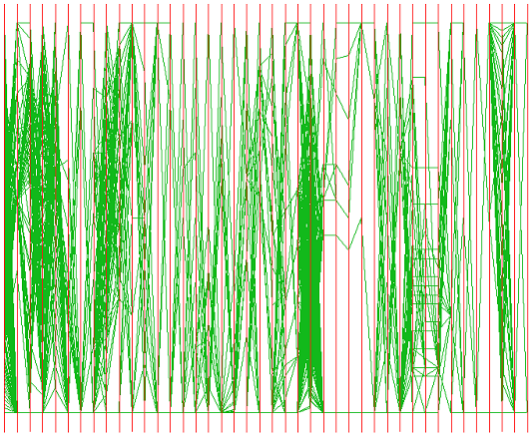

Figure 1.2: A subset of Census Income data set (42 dimensions, 200 data items) in Parallel Coordinates. Individual data items cannot be seen clearly.

To overcome these problems, one promising approach is dimensionality reduction [11]. This idea is to first reduce the dimensionality of the data, and then to visualize the data set in the reduced dimensional space. There are currently three popular dimensionality reduction techniques used in data visualization. Principal Component Analysis (PCA) [12] attempts to project data into a few dimensions that account for most variance in the data. Multidimensional Scaling (MDS) [13] is an iterative non-linear optimization algorithm for projecting multidimensional data down to a reduced number of dimensions. Kohonen’s Self Organizing Map (SOM) [14, 15] is an unsupervised learning method to reduce multidimensional data to 2D feature maps [16]. There are several visualization systems that have adapted one or more of these dimensionality reduction techniques [17, 18, 16].

There are several inherent serious drawbacks in the existing dimensionality reduction

techniques. One is that after the projection from the high dimensional space to a low dimensional space, the original data values are lost and the dimensions in the low dimensional space would likely have no clear meaning to the users. As noted by Brodbeck [16], such techniques create a space with no inherent meaning other than the overall information relationship reflected in the collection [19]. Another drawback, although not as obvious, is that little interaction is allowed during those highly automatic processes besides adjusting weight factors. Often specialists in their field would understand the meaning of the dimensions better than a general reduction technique could infer from the data set. Such domain experts would know which dimensions are more important than others. It is a major deficiency of current techniques that such "human intelligence" is not effectively used in a dimension reduction approach.

A flexible approach for dimensionality reduction that generates meaningful subspace and allows user interactions is needed in the field of multidimensional visualization. The design and development of such an approach is the goal of this thesis.

1.3 Goals of This Thesis

In this thesis, we intend to explore a new approach to dimensionality reduction that generates meaningful subspaces, allows users to interact in an interactive, and visual way, and is compatible with most of the existing multidimensional display techniques, such as parallel coordinates, star glyphs and scatterplot matrices. We call our new approach for the reduction and exploration process "visual hierarchical dimension reduction" (VHDR).

The goals of VHDR are:

- to scale with data sets containing up to tens of thousands dimensions;
- to reduce the dimensionality of high dimensional data sets without losing without losing the major information the data sets contained and generate "meaningful" low

dimensional subspaces in a multi-resolution manner;

- to allow users to take an active part in this dimension reduction approach in an interactive and visual way, while avoid tedious manual operations;

- to be compatible with most of the existing multidimensional display techniques, such as parallel coordinates, star glyphs and scatterplot matrices.

1.4 Overview of Our Approach: Visual Hierarchical Dimension Reduction

Our visual hierarchical dimension reduction approach organizes all dimensions of a high dimensional data set into a dimension hierarchy, generates lower dimensional subspaces using this hierarchy, and displays the mapping of the data set in the lower dimensional subspaces using existing multidimensional display techniques. This approach can be divided into three steps:

- Step 1: Dimension Clustering

  In this step, we organize all the original dimensions of a multidimensional data set into a hierarchical dimension cluster tree according to the similarities among the dimensions. Each original dimension is mapped to a leaf node in this tree. Similar dimensions are placed together and form a new cluster, and similar clusters in turn compose higher-level clusters. The tree is a hierarchy of the clusters, with the leaves being clusters composed of one individual dimension only.

  This dimension clustering process is an automatic process with user controlled clustering parameters. Users can provide their own similarity calculation routines instead of the system-provided similarity calculation routines. We also allow users to create the hierarchical dimension cluster tree manually.

A representative dimension is assigned or generated for each dimension cluster. Several options of how to assign or generate the representative dimensions are provided. Users can either select one of them or provide their own options.

- Step 2: Interactive Dimension Hierarchy Visualization and Selection

  In this step, users can visually explore the generated hierarchical dimension cluster tree and select some dimension clusters from the tree in order to form a lower dimensional subspace. The hierarchical dimension cluster tree is visualized in a radial, space-filling display called an "Interactive Sunburst"(Sunburst). Users are allowed to interactively navigate and modify the hierarchical dimension cluster tree in Sunburst, and interactively select clusters from the hierarchical dimension cluster tree. Several brush mechanisms are provided to users to facilitate their dimension cluster selection.

- Step 3: Data Visualization in Lower Dimensional Subspace

  In this step, we map the data set from the high dimensional space to a lower dimensional subspace composed of the representative dimensions of the clusters selected by users in Step 2. Then we visualize the mapped data set in the lower dimensional subspace using the existing multidimensional visualization techniques to get reduced dimensionality displays. Users can generate different lower dimensional subspace by selecting different sets of dimension clusters from the hierarchical dimension cluster tree.

  In the reduced dimensionality displays, some dimensions are representations of clusters of dimensions. Users may be interested in the degree of dissimilarity within the dimension clusters. We provide several options to visualize the dissimilarity information in the reduced dimensionality displays.

1.5 Thesis Organization

Recent research regarding visualization of high dimensional data sets, visualization of hierarchies, distortion techniques and clustering algorithms are surveyed in Chapter 2. Chapter 3 gives an overview of the XmdvTool system. Details of our VHDR approach are presented in Chapters 4, 5 and 6. Chapter 7 describes our implementation. Chapter 8 presents two case studies. Conclusions and open issues for future work are described in Chapter 9.

Chapter 2

Related Work

2.1 Visualization Techniques for High Dimensional Data Sets

Most traditional multi-dimensional visualizations become cluttered when the dimensionality of a data set it displays is high. Here we discuss some visualization approaches intended to handle high-dimensional data sets.

2.1.1 Dimension Reduction Approaches

A natural way to handle high-dimensional data sets is to reduce their dimensionality so they can be fit into traditional visualization techniques. There are three major approaches to dimensionality reduction. Principal Component Analysis (PCA) [12] attempts to project data down to a few dimensions that account for most variance within the data. Multidimensional Scaling (MDS) [13] is an iterative non-linear optimization algorithm for projecting multidimensional data down to a reduced number of dimensions. Kohonen's Self Organizing Map (SOM) [14, 15] is an unsupervised learning method to reduce multidimensional data to 2D feature maps [16].

Recently, many new dimensionality reduction techniques have been proposed to process large data sets with relatively high dimensionality. For example, Random Mapping [20] projects the high dimensional data to a lower dimensional space using a random transform matrix. [20] presented a case study of a dimension reduction from a 5781-dimensional space to a 90-dimensional one with Random Mapping. Anchored Least Stress [21, 22] combines PCA and MDS and makes use of the result of data clustering in the high dimensional space so that it can handle very large data sets. We take their idea in our dimension clustering approach by using data clusters as input instead of original data items.

There are many visualization systems that make use of existing dimensionality reduction techniques [18, 16, 19]. Galaxies and ThemeScape [18] project high dimensional document vectors and their cluster centroids down into a two dimensional space, and then use scatterplots and landscapes to visualize them [22]. Bead [16] uses MDS to lay out high dimensional data in a two dimensional or three dimensional space and uses imageability features to visualize the data.

All the above mentioned approaches have the common drawback that their generated display spaces typically have no clear meaning for the users. In our approach, we reduce the dimensionality in an interactive manner so as to generate a meaningful low dimensional subspace.

2.1.2 Grouping, Clustering and Reordering Dimensions Approaches

Worlds within worlds [23] visualizes a high dimensional data set in 3D space by grouping the dimensions into triple sets. Each triple set of dimensions form a 3-D space. The first set of dimensions form the outmost 3-D space. By selecting a point in it, users can enter the 3-D space composed of the second set of dimensions. Recursively, users can go

through all the dimension sets one by one.

One group of researchers [24] have proposed a clustering method to layout a $m * n$ multi-dimensional matrix composed of n variables and m data items. They rearrange the matrix in such a way that the data items that are similar are close to each other, while similar variables are also positioned close to each other. We go further by applying the clustering approach into dimension reduction.

Ankerst et al. [25] use similarity clustering of dimensions to order and arrange dimensions in a multidimensional display. They arrange the dimensions so that dimensions showing a similar behavior are positioned next to each other. Their work gave us the hint that similarity among the dimensions is an important parameter that can be used to group the dimensions.

2.2 Visualization Techniques for Large-Scale Multidimensional Data Sets

There are many approaches towards visualizing large-scale multi-dimensional data sets, such as pixel-oriented techniques (including spirals [26], recursive patterns [9], and circle segments [27]), multiresolution multidimensional wavelets [28], pixel bar charts [1], and Interactive Hierarchical Displays [29, 30].

Hierarchical Parallel Coordinates [31] is one of the Interactive Hierarchical Displays [30] developed for visualizing large multidimensional data sets in the context of our XMDV project. Since displaying a large number of data items will clutter the screen, Hierarchical Parallel Coordinates group the data items into a hierarchical cluster tree. A set of clusters selected from a certain level of detail in the hierarchical cluster tree is visualized on the screen instead of all the data items in the data set. The clusters are visualized by center lines and bands which respectively represent the mean points and extents of the clusters. Through interactive operations such as roll-up and drill-down, users can visu-

ally select the set of clusters to be shown on the screen at their preferred level of detail. Moreover, users can select part of the clusters displayed on the screen to be highlighted or masked. The same framework used to develop Hierarchical Parallel Coordinates has been also applied to Hierarchical Scatterplot Matrices, Hierarchical Star Glyphs, and Hierarchical Dimensional Stacking. Details of these can be found in [30].

Hierarchical Parallel Coordinates group the data items into a hierarchical cluster tree to avoid the clutter problem. In our approach of reducing dimensionality, which is aimed at avoiding clutter caused by a large number of dimensions rather than a large number of data items, we instead group the dimensions into a hierarchical dimension cluster tree. Although at first glance these two approaches appear to be completely different, there is actually a large overlap of strategies that can be brought to bear to solve these two problems. As a result, we can reuse some concepts of Hierarchical Parallel Coordinates in our approach. In particular, we adapt interactive operations such as structure-based brushing and roll-up/drill-down for our navigation of the hierarchical dimension cluster tree.

2.3 Visualization Techniques for Hierarchies

How to display a hierarchical information structure, or, in short, trees, is a problem that has been widely studied. The main difficult in tree visualization is to visualize the structure of large trees efficiently. For example, using the best known file browsers, it can be hard to form a mental image of the overall structure for large trees [32]; traditional node and link diagrams [33], in which elements are shown as nodes and relations are shown as links from parent to child nodes, use the display space inefficiently and in general are only effective for very small trees.

There are many tree visualization methods aimed at visualizing large trees, such as

improved node and link diagrams [34, 35], cone trees [36, 37] and their variations [38, 39, 40], collapsible cylindrical trees [41], botanical trees [42], radial layouts [43, 44, 45, 46], hyperbolic layouts [47, 48, 49, 50, 51], skeletal images [45], and castles [52]. Among them, space-filling techniques [53, 54, 55]are very popular. Among the space-filling techniques, there are rectanguler space-filling techniques, such as treemaps and its variations [53, 54, 55, 56, 57, 58], and radial (also called circular) space-filling techniques [56, 57, 58].

A treemap [53, 54] is constructed via recursive subdivision of the initial rectangle. It uses the display space very efficiently when sizes of the nodes are the most important feature to be displayed. It has a major disadvantage in that it falls short in visualizing the structure of the tree as clearly as some of the other methods [55]. Figure 2.1 shows a treemap of the Iris data set generated by XmdvTool. Nested treemaps [54] improve treemaps by using a slightly smaller rectangle instead of the initial rectangle during the subdivision process to view the hierarchical structure better. However, the effective display space is reduced at the same time. Cushion treemaps [55] improve the structure-revealing ability through another approach. In a cushion treemap, ridges are added to the rectangles during the subdivision, which are rendered with a simple shading model. The result is a surface that consists of recursive cushions. Figure 2.2 is a cushion treemap from the cushion treemap paper [55].

Recent user studies indicate that radial space-filling techniques work better in revealing hierarchical structures than treemaps [59, 60], while also making efficient use of the display space. Sunburst [58] is an example of the radial space-filling hierarchy visualization technique. In Sunburst, deeper nodes of the hierarchy are drawn further from the center and child nodes are drawn within the arc subtended by their parents. The angle occupied by a node is proportional to its size.

Radial space-filling techniques have some advantages over other tree drawing strate-

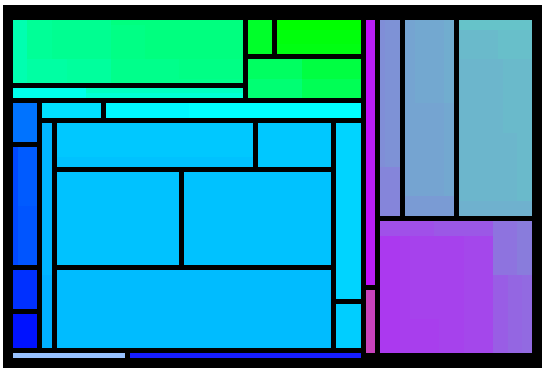

Figure 2.1: Treemap

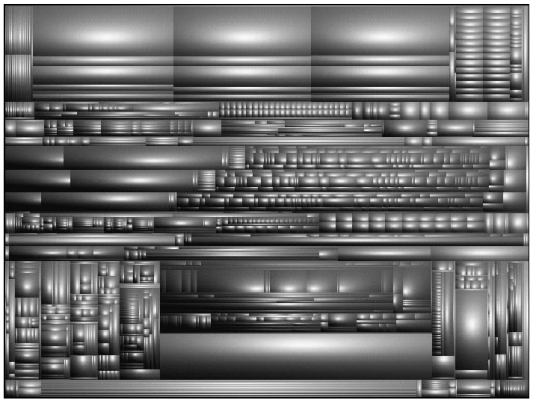

Figure 2.2: Cushion Treemap from [55]

gies. First, as one of the space-filling techniques, they use more implicit containment and geometry characteristics to present a hierarchy than tree drawing algorithms. The later utilize edges between nodes to indicate parent-child structure [58]. Second, compared to treemaps, radial space-filling techniques are better in conveying the hierarchy structure [56, 57, 58].

Radial space-filling techniques have a drawback, the small slices are difficult to distinguish. This drawback can be overcome by using "context+focus" techniques. Andrews and Heidegger's radial space-filling system [56] uses two semi-circular areas as a form of two-level "overview and detail" [58]. Stasko and Zhang felt that this was not smooth and flexible enough in alternating between global and detailed views [58]. They proposed three distinct distortion methods to solve this problem: angular detail method, detail outside method, and detail inside method. The angular detail method shrinks the entire hierarchy and pushes it aside. The selected area is then enlarged and put in the center of the display. While this looks natural to the users, it does waste space. The detail outside method shrinks the entire hierarchy in the center. The selected nodes are expanded to be a new complete circular ring-shaped region around the overview [58]. The detail inside method pushes the entire hierarchy outward and shows the selected items inside the entire hierarchy. Using this method the item in the overview can be viewed clearer.

We have selected Sunburst to visualize our hierarchical dimension cluster tree because of its advantages mentioned above. However, our Sunburst is different from Stasko and Zhang's Sunburst in several ways. In particular, it is different in that:

- we use color to convey structural information of the hierarchy displayed, while Stasko and Zhang use color to convey other information, such as file attributes in a file hierarchy;
- we propose our angular distortion technique to view details within context. It allows multi-focus distortion and is different from the existing distortion techniques for the radial space-filing hierarchical visualization;
- we allow users to interactively modify the hierarchy directly from the Sunburst display;
- we provide brushing mechanisms to the users to allow them select clusters from the hierarchy.

2.4 Clustering Algorithms

For the purpose of constructing a hierarchical dimension cluster tree, we need to first cluster the dimensions. Our idea is to adapt one of the existing data point clustering algorithms to the dimension clustering problem.

There are two basic types of clustering algorithms [61, 62]: partitioning [63, 64] and hierarchical algorithms [65, 66, 67]. Partitioning algorithms divide all the data points into a given number of clusters, while hierarchical algorithms construct a hierarchical cluster tree by recursively splitting the data set into smaller clusters until every leaf cluster contains only one or a few data points.

The k-Means algorithm is a popular partitioning algorithm. It picks k cluster centroids and assigns points to the clusters by picking the closest centroid to the point in question. The centroids of the clusters may shift when new points are added into clusters so the process may need to be repeated. BFR [64] is an algorithm based on k-Means algorithm. It intends to cluster large data sets that cannot be loaded into the main memory at one time by identifying regions of the data that are compressible, regions that must be maintained in memory, and regions that are discardable. This algorithm works best if the clusters are normally distributed around some central points.

CURE [65] is a sampling-based hierarchical clustering algorithm for large data set. Compared with k-means approaches, which work well only for clusters that are neatly expressed as Gaussian noise around a central point, CURE is more robust in that it is able to identify clusters having non-spherical shapes and wide variances in size. CURE achieves this by representing each cluster by a certain fixed number of points that are generated by selecting well scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. Having more than one representative point per cluster allows CURE to adjust well to the geometry of non-spherical shapes and the shrinking helps to dampen the effects of outliers. To handle large databases, CURE employs a combination of random sampling and partitioning. A random sample drawn from the data set is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters.

BIRCH [67] is another efficient clustering algorithm for large data sets. In the BIRCH algorithm, objects are read from the database sequentially and inserted into incrementally evolving clusters that are represented by generalized cluster features (CFs). A new object read from the database is inserted into the closest cluster, an operation which potentially requires an examination of all existing CFs. Therefore BIRCH organizes all clusters in an in-memory index, a height-balanced tree called a CF-tree. For a new object, the search for

an appropriate cluster now requires time logarithmic in the number of clusters as opposed to a linear scan.

For high dimensional space, it is common that clusters only exist in some subspaces. CLIQUE [68] is a clustering algorithm that is able to find clusters embedded in subspaces of high dimensional data. It identifies dense clusters in subspaces of maximum dimensionality. It generates cluster descriptions in the form of DNF expressions that are minimized for ease of comprehension.

In this thesis work we need to construct dimension hierarchies from a high dimensional data sets. We achieve this by clustering the dimensions. We can adapt existing data clustering algorithm for dimension clustering. But in the dimension clustering, the roles of data points and dimensions have to be interchanged. From the view of a data clustering algorithm, the dimension clustering works with relatively fewer data points (the number of dimensions that need to be clustered) while in a much higher dimensional space (the number of data points used to cluster the dimensions) since usually the latter is much larger than the former. In such a high dimensional space, data is sparse in nature and clusters only exist in subspace. Thus we need an approach that can find clusters in subspaces. Moreover, since we want a multi-resolution exploration of the dimension hierarchy, our dimension clustering must be bend on a hierarchical approach towards clustering.

Chapter 3

Background on the XmdvTool

3.1 Overview

To prove that our VHDR approach is practical and compatible with existing multidimensional display techniques, we built a VHDR prototype as an extension to the XmdvTool system developed by Ward et al. at WPI [69, 70, 29, 71, 72, 30]. XmdvTool is a public-domain software package for interactive visual exploration of multivariate data sets. It is available on all UNIX platforms which support XR4 or higher. XmdvTool 4.0 and later versions are also available on Windows95/98/NT platforms, and are based on OpenGL and Tck/Tk. It supports four methods for displaying both (non-hierarchical) flat form data and hierarchically clustered data, namely scatterplots, star glyphs, parallel coordinates, and dimensional stacking. XmdvTool also supports a variety of interaction modes and tools, including brushing in screen, data, and structure spaces, zooming, panning, and distortion techniques, and the masking and reordering of dimensions. Univariate displays and graphical summarizations, via tree-maps and modified Tukey box plots, are also supported. Finally, color themes and user customizable color assignments permit tailoring of the aesthetics to the users. XmdvTool has been applied to a wide range of application

areas, such as remote sensing, financial, geochemical, census, and simulation data.

We introduce the existing functions of XmdvTool briefly below. To learn detailed information about the flat visualization and N-dimensional brushing techniques, see [69, 70]. To learn detailed information about the hierarchical visualization techniques and their interactive tools in XmdvTool, see [29, 71, 72, 30]. More information about the XmdvTool project can be obtained from `http://davis.wpi.edu/~xmdv`.

3.2 Flat Visualization Techniques in XmdvTool

3.2.1 Flat Visualization Techniques

XmdvTool supports four methods for displaying flat form data, namely Parallel coordinates (see Figure 3.1), star glyphs (see Figure 3.2), scatterplot matrices (see Figure 3.3), and dimensional stacking (see Figure 3.4).

In parallel coordinates [6, 7], each dimension is represented as a uniformly spaced vertical axis. A data item in this multidimensional space is mapped to a polyline that traverses across all the axes. Figure 3.1 shows the Iris data set (4 dimensions, 150 data items) using parallel coordinates.

In star glyphs, each data item is represented by an individual shape [2, 4, 5]. In a star glyph, the data values are mapped to the length of rays emanating from a central point, and the ends of the rays are linked to form a polygon. We can view these rays as axes, with each axis representing a dimension. The directions of these axes are from the center point to the outside. Figure 3.2 shows the Iris data set using star glyphs.

In scatterplot matrices, each data item is projected to $N * N$ plots, with N being the number of dimensions of the data set. The position of the projected point in a plot is decided by the values of the data item in the two dimensions that compose this plot. Figure 3.3 shows the Iris data set using scatterplot matrices.

Flat dimensional stacking [10] displays an N dimensional data set by recursively embedding pairs of dimensions within one another. Each dimension is discretized into a small number of subranges, and two dimensions are initially selected to subdivide the display space into subimages whose size depends on the number of subranges or bins used for those dimensions. These subimages are subdivided based on the next two dimensions, and the process repeats until all dimensions have been mapped. Thus the multivariate space is split into a number of small cells that each map to a segment of the screen. Each data item will fall into one of these small cells, and thus have an assigned screen position. Figure 3.4 shows the Iris data set using dimensional stacking.

3.2.2 Brushing in Flat Visualizations

XmdvTool has implemented N-D brushes in the flat visualizations [70] to allow users to select subsets of data items from the visualized data sets. Many useful operations, such as highlighting, magnifying, or saving as new data sets, can be applied to the selected subsets. N-D brushes have the following characteristics:

- N-D hyperbox shape
- step edge or ramp edge boundary
- boolean operations among multiple brushes
- multiple methods to control the sizes and positions of the brushes

XmdvTool4.1 provides a function for saving brushed data. This function will save the brushed data as a new file in the XmdvTool format.

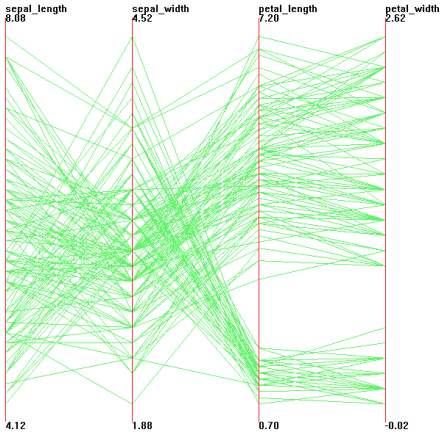

Figure 3.1: Iris Data Set in Flat Parallel Coordinates

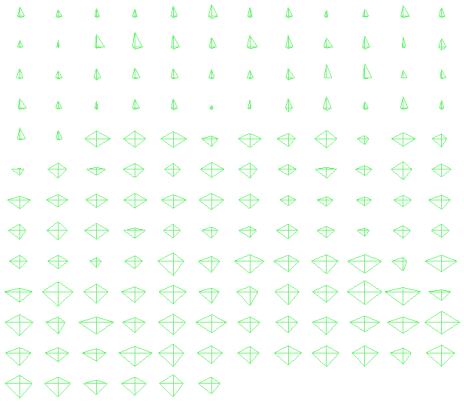

Figure 3.2: Iris Data Set in Flat Star Glyphs

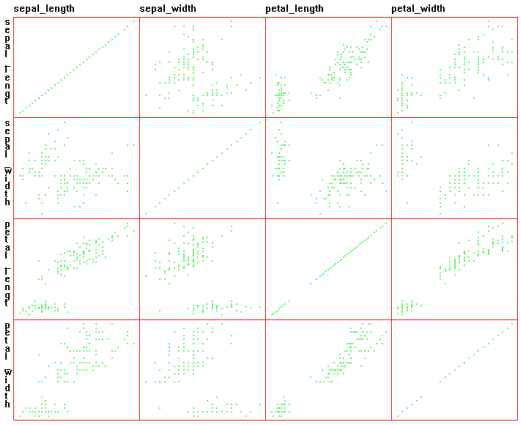

Figure 3.3: Iris Data Set in Flat Scatterplot Matrices

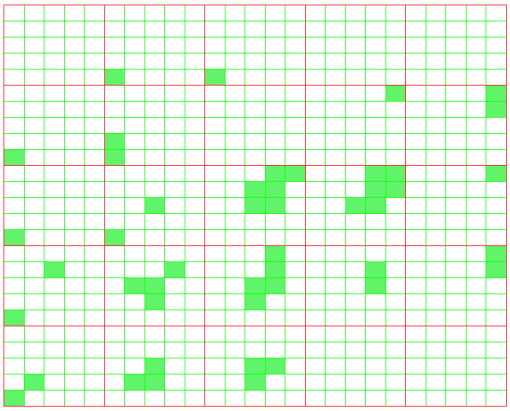

Figure 3.4: Iris Data Set in Flat Dimensional Stacking

3.3 Hierarchical Data Analysis in XmdvTool

3.3.1 Hierarchical Visualization Techniques

The flat visualizations become very crowded when they are applied to large-scale data sets. To overcome this clutter problem, we have developed an Interactive Hierarchical Display framework [30]. The underlying principle of this framework is to develop a multi-resolution view of the data via hierarchical clustering, and to use hierarchical variations of

traditional multivariate visualization techniques to convey aggregation information about the resulting clusters. Users can then explore their desired focus regions at different levels of detail, using our suite of navigation and filtering tools.

By applying this framework to the flat visualization techniques, we extent XmdvTool with hierarchical parallel coordinates (see Figure 3.5), hierarchical star glyphs (see Figure 3.6), hierarchical scatterplot matrices (see Figure 3.7) and hierarchical dimensional stacking (see Figure 3.8).

Hierarchical parallel coordinates are an extension of traditional (flat) parallel coordinates. In hierarchical parallel coordinates, the clusters replace the data items. The mean of a cluster is mapped to a polyline traversing across all the axes, with a band around it depicting the extents of the cluster in each dimension. The lower edge of the band intersects each axis at the minimum value of its respective cluster in that dimension. The upper edge of the band intersects each axis at the maximum value of its respective cluster in that dimension. To give the user a sense of the location of data points in a cluster and to convey the overlap among clusters, each band is translucent. We assume that there is a linear drop-off in the density of cluster data from its center to the edge, and set the maximum opacity proportional to the population.

Hierarchical star glyphs are an extension of flat star glyphs. In hierarchical star glyphs, each star glyph represents a cluster. The mean values are used to generate the basic star shape. The band around the mean polygon has two edges; one is outside the mean polygon and another one is inside the mean polygon. The inside edge intersects each axis at the minimum value of its respective cluster in that dimension, while the outside edge intersects each axis at the maximum value of its respective cluster in that dimension. Obviously, if we draw a star glyph starting from the same center point to present a data item included in that cluster, this star glyph would be inside the band of that cluster. Thus the band successfully depicts the extent of the cluster.

Hierarchical scatterplot matrices are an extension of flat scatterplot matrices. The mean of a cluster is drawn as an ordinary data item in flat scatterplot matrices. The extents of each cluster form rectangles around the projected mean in each plot. The projections of a cluster on different plots are drawn in the same color, which gives users the possibility to more easily link a cluster from one plot to another. In the flat form scatterplot matrices, all the data items have the same color. Hence users can have difficulty linking a data item when they move from one plot to another, although selective highlighting helps this linkage.

Hierarchical dimensional stacking is an extension of flat dimensional stacking. The clusters replace the data items. The mean of a cluster will fall into a single small block as if it were an original data item in the flat form dimensional stacking. The band of this cluster depicting the cluster extents may potentially map to many blocks. This time it is possible that some parts of the band are disjoint from others due to the embedding process, even though they are adjacent in N-dimensional space.

3.3.2 Interactive Tools in Hierarchical Visualizations

There are several interactive tools, such as the structure-based brush, drill-down/roll-up operations, extent scaling, and dynamic masking, to interactively explore the hierarchical visualizations. These are described briefly below. Details can be found in [71, 31, 72, 30].

- Structure-based Brush

  A structure-based brush allows users to select subsets of a data structure by specifying focal regions as well as a levels-of-detail on a visual representation of the structure.

- Drill-down/Roll-up Operations

Drill-down/roll-up operations allow users to change the level of detail of interactive hierarchical displays intuitively and directly. Drill-down refers to the process of viewing data at an increased level of detail, while roll-up refers to the process of viewing data with decreasing detail [29]. The drilling operations are coupled with brushing. XmdvTool permits selective drill-down/roll-up of the brushed and non-brushed region independently.

- Extent Scaling

  Extend scaling solves the problem of overlapping among the bands by decreasing the extents of all the bands in each dimension by scaling them uniformly via a dynamically controlled extent scaling parameter. Users can still differentiate between clusters with large and small extents after the bands have been scaled.

- Dynamic Masking

  Dynamic masking refers to the capability of controlling the relative opacity between brushed and unbrushed clusters. It allows users to deemphasize or even eliminate brushed or unbrushed clusters. With dynamic masking, the viewer can interactively fade out the visual representation of the unbrushed clusters, thereby obtaining a clearer view of the brushed clusters while maintaining context regarding unbrushed areas. Conversely, the bands of the brushed clusters can be faded out, thus obtaining a clearer view of the unbrushed region. Used together with the structure-based brush, dynamic masking reduces the overlapping and density of the clusters on the screen by fading out uninteresting clusters so that users can concentrate on the clusters of interest.

3.4 Common Interactive Tools Used in Both Flat and Hierarchical Visualizations

There are some interactive tools in XmdvTool that can be used in both the flat and hierarchical displays. There are briefly described below:

- Zooming and Panning

  The whole display area can be zoomed and panned.

- Dimension Distortion

  In Parallel Coordinates, the distance between two adjacent axes can be increased or decreased. In Scatterplot Matrices, the size of a single plot can be enlarged or reduced. In Star Glyphs, the angle between two adjacent arms can be increased or decreased.

- Dimension Enabling/Disabling and Reordering

  Dimensions can be enabled/disabled and reordered.

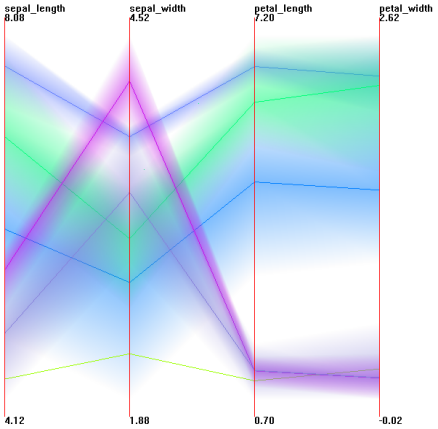

Figure 3.5: Iris Data Set in Hierarchical Parallel Coordinates

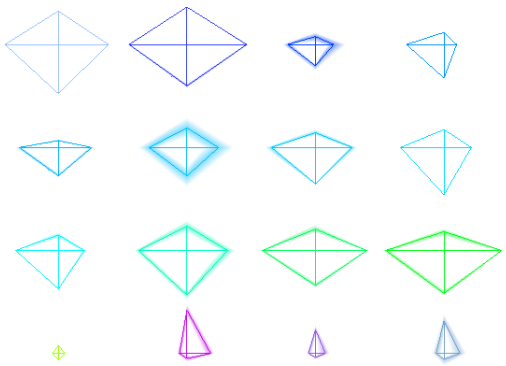

Figure 3.6: Iris Data Set in Hierarchical Star Glyphs

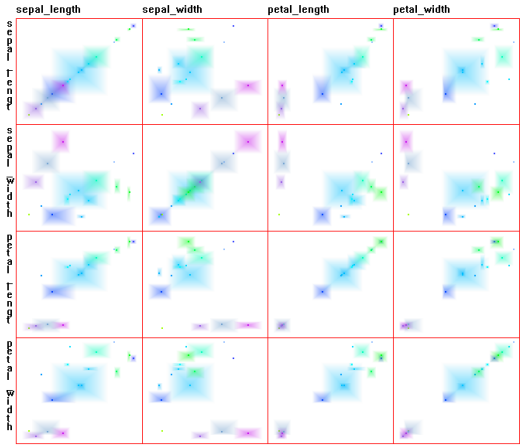

Figure 3.7: Iris Data Set in Hierarchical Scatterplot Matrices

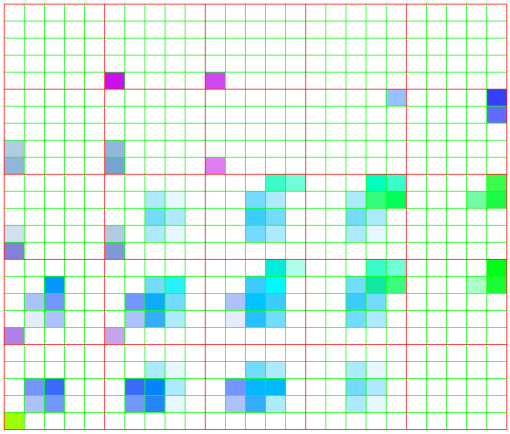

Figure 3.8: Iris Data Set in Hierarchical Dimensional Stacking

Chapter 4

Dimension Clustering and Representative Dimensions

4.1 Dimension Clustering

Dimension clustering is the first step in the VHDR approach. The dimension hierarchy generated by the dimension clustering step plays an important role in the overall VHDR process. One of the most important assumptions of VHDR is that if dimensions are placed into one cluster of the dimension hierarchy, then they are more similar to each other than to any dimensions outside that cluster. Given this assumption, we argue that we can use a representative dimension to represent all the dimensions in a cluster and that we can visualize the data set in a subspace composed of the representative dimensions without losing too much information.

The goal of the dimension clustering step is to efficiently generate an acceptable dimension hierarchy even when there are thousands dimensions and millions of data items. This dimension hierarchy should be able to reflect the similarities among the dimensions as accurately as possible. Online dimension clustering is required since we want to give the users the flexibility of regenerating the dimension hierarchy within an acceptable response time after the users enable or disable some dimensions. Finally, since most high dimensional data sets also have large numbers of data items, we need to maintain the

effectiveness and efficiency of the dimension clustering algorithm even if the data set is very large.

4.1.1 Overview of Our Dimension Clustering Algorithm

The basic idea of our dimension clustering algorithm is to cluster the dimensions iteratively. In each iteration, the clusters generated should have dissimilarities less than or equal to a threshold dissimilarity (this threshold dissimilarity increases from 0 to 1 gradually from the first iteration to the last iteration). First, we compare each pair of the "candidate" dimensions (all the original dimensions in the first iteration) and record the dimension pairs that have dissimilarities smaller than or equal to the threshold dissimilarity as "similar pairs" in a similar pairs array. After that, we determine the dimension that forms the maximum number of similar pairs. We label this a cluster center. We include all the dimensions that formed similar pairs with the cluster center into this cluster. Once a dimension is included into a cluster, we delete all the similar pairs that include that dimension from the similar array list. We repeat this process until there are no similar pairs. Then we generate a representative dimension for each generated cluster(see Section 4.2). The representative dimension of the newly formed clusters and the candidate dimensions that did not get included in any newly formed clusters are the input for the next iteration. If a representative dimension is included in a cluster, it means that the cluster it represents is a child of that cluster. The threshold dissimilarity is increased in the next iteration. Started from the first iteration with all the original dimensions as the candidate dimensions, the process will end up when all the original dimensions have been grouped into one cluster. The threshold dissimilarity of 1 at the last iteration makes sure that the process will terminate.

4.1.2 Data Issue: How To Handle Large-Scale Data Sets

When we calculate the dissimilarity of two dimensions, a natural method to do this is to use all the data items in the original data set. However, for a large data set, if we use all of the data items to calculate the dissimilarities, the calculations would be slow and the memory required would be large. We hence look for a way to improve this situation. Our solution is to use a data hierarchy of the large data set (it has been built in our hierarchical visualizations) to perform the dimension clustering, as explained below.

First, we select some data clusters from the data hierarchy using the structure-based brush in the hierarchical visualizations [71]. In the data hierarchy, every data cluster has a radius and the distance between any two data items in a cluster is less than its radius multiplied by two. In the selection, a threshold radius is given and all the data clusters selected must have radii less than or equal to the threshold radius. We limit the radii of the clusters selected since this would have an effect on the dissimilarity calculation (as discussed in Section 4.1.3). Each leaf node in the data hierarchy should be covered exactly once in the selected data clusters so that every data item is counted in the dissimilarity calculation.

Second, we use these selected data clusters instead of the original data items to calculate the dissimilarity between dimensions (see Section 4.1.3 for details). In the dissimilarity calculation, the means of the data clusters (averages of all their data items), the cluster radii and the cluster entries (the number of data items included in a cluster) are used.

By using data clusters instead of original data items, we accelerate the dissimilarity calculation and reduce the memory requirement. For example, assume a dissimilarity calculation between two dimensions on n data items has a time complexity of $o(n^2)$. By using $n/2$ data clusters instead of n data items, the time complexity can be reduced to $o(n^2/4)$.

However, there is a tradeoff between the efficiency and accuracy. The larger the

threshold radii, the higher the efficiency, but the lower the accuracy (we will discuss it in detail in Section 4.1.3). In our system, the threshold radius is adjustable. When the threshold radius is equal to zero, it means that all the original data items in the data set are used in the dissimilarity calculations.

4.1.3 Dissimilarity Calculation

The lion's share of our dimension clustering algorithm is performing dissimilarity calculations. In section 4.1.2, we show how we reduce the complexity of the dissimilarity calculation by using a smaller number of data clusters instead of all the original data items. Here we discuss how the data cluster parameters affect the dissimilarity calculation:

- entries: a data cluster with many members means that it represents more original data items than a data cluster with fewer members. Hence the former should have more contribution to the calculated dissimilarity than the latter.

- radius and mean: In Figure 4.1, $d1$ and $d2$ are the means of a data cluster projected to dimension 1 and dimension 2 (suppose $d1 > d2$). Suppose that the data cluster has a radius r, then the projections of all the data items in the data cluster on dimension 1 and dimension 2 will be within the ranges: $[d1-r, d1+r]$ and $[d2-r, d2+r]$. Thus the maximum possible dissimilarity of all the original data items in this data cluster max_dis is $d1 - d2 + 2 * r$. Similarly, the minimum possible dissimilarity of all the original data items in this data cluster min_dis is $d1 - d2 - 2 * r$. Hence the average dissimilarity of all the original data items in this data cluster ave_dis is in the range: $[d1 - d2 + 2 * r, d1 - d2 - 2 * r]$. When r is much smaller than $d1 - d2$, we can approximate that $ave\_dis = d1 - d2$. It means that the threshold radius for the data cluster selection should be much smaller than the dissimilarity threshold used in the dimension clustering algorithm.

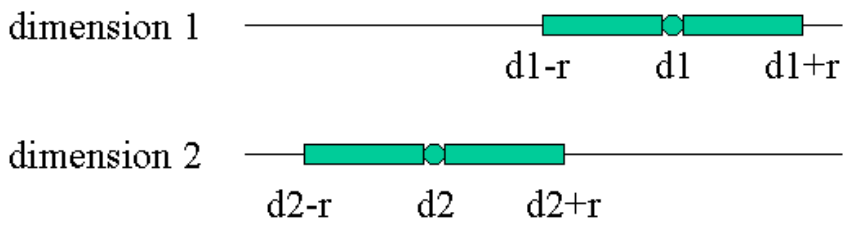

Figure 4.1: Effect of Radius and Mean of Data Cluster on Dissimilarity Calculation: $d1$ and $d2$ are the means of a data cluster projected to dimension 1 and dimension 2 (suppose $d1 > d2$). The data cluster has a radius r. The projections of all the data items in the data cluster on dimension 1 and dimension 2 will be within the ranges: $[d1 - r, d1 + r]$ and $[d2 - r, d2 + r]$.

Since we need to keep the threshold radius relatively small, the number of selected data clusters could be still very large when the size of the data set is extremely large. Thus we need to view the speed of the dissimilarity calculations as one important factor when we choose our approach to the dissimilarity calculations.

The most commonly used dissimilarity calculation statistics model, Pearson's R, is computationally costly, and many other more complicated statistics models exist. Hence we did not use them. However, we allow users to plug in Pearson's R or other approaches as a substitute for our own approach.

We have the following considerations in our approach:

- A data cluster's dissimilarity between two dimensions is calculated as the distance between the normalized means of the two dimensions.

- There are always outliers in the data sets. We consider that two dimensions have a dissimilarity $thre\_dis$ if for certain percentage of the data items, their projections on these two dimensions have differences less than or equal to $thre\_dis$. For this sake a percentage threshold is given. For example, we can stipulate that if there

exist more than 90% of the data items whose projections on these two dimensions have differences less than or equal to $thre\_dis$, dissimilarity of the two dimensions is $thre\_dis$. Here the percentage threshold is 90%. The percentage threshold can be adjusted by users.

- Assume there are three dimensions A, B and C. Assume A and B have a dissimilarity that is less than $th1$ at the percentage threshold 90%, and A and C have a dissimilarity of $th2$ at the percentage threshold 90%. Since we know that there are at most 10% data items whose projections on A and B have differences larger than $th1$ and there are at most 10% data items whose projections on A and C have differences larger than $th1$, we know that there are at least 100%−10%−10%= 80% data items have projections on B and C less than or equal to $th1 + th2$. Thus B and C have a dissimilarity of $th1 + th2$ at the percentage threshold 80%. Generally, if every dimension in a dimension cluster has a dissimilarity of $thre\_dis$ at the percentage threshold X% with a certain dimension (cluster center), then each pair of dimensions in the dimension cluster has a dissimilarity under $2 * thre\_dis$ at the percentage threshold $100 - 2 * (1 - X)$%. Hence the maximum dissimilarity and the percentage threshold for a dimension cluster is controllable by adjusting the two parameters $thre\_dis$ and X.

Figure 4.2 is the algorithm to judge if two dimensions ($i1$ and $i2$) have a dissimilarity $thre\_dis$ given the percentage threshold and the size of the data set. $thres\_dataitems$ is defined as the percentage threshold multiplied by the size of the data set.

4.1.4 Alternative Approaches for Dimension Clustering

Other approaches to achieving dimension clustering are possible. For example, we can use factor analysis [73] to form a dimension hierarchy since it is able to group dimensions

```
Bool JudgeTwoDimensionsDissimilarity(dimension d1, dimension d2,
                        double thres_dis, int thres_dataitems)
Begin
   int count = 0;
   For every data cluster C_i
      Begin
         If (fabs(C_i.data[d1] − C_i.data[d2]) < thres_dis)
            count+ = C_i.entries;
         Endif
      End
   If ( count > thres_dataitems )
      Return true;
   Else
      Return false;
   Endif
End
```

Figure 4.2: Algorithm of Judging Two Dimensions' Dissimilarity

according to their correlations. In the future, these and other different approaches should all be incorporated into our system in order to be compared.

4.2 Representative Dimensions: Assign or Create?

Having built the hierarchical dimension cluster tree, we need to assign or create a representative dimension for each dimension cluster so that it can be used to represent the dimensions in its dimension cluster in the multi-dimensional displays. Below we propose several possible approaches. In the description, if we mention original dimensions composing a dimension cluster, we mean all the leaf nodes included in this dimension cluster.

4.2.1 Approach 1: Averaging the Original Dimensions

This approach uses the average of the original dimensions in a dimension cluster as the representative dimension. It means that when we calculate a data item's projection to the representative dimension, the projection is equal to the average of the projections of these data items to all the original dimensions composing this dimension cluster. This approach seems a fair solution to the original dimensions since every original dimension equally contributes to the representative dimension. This is the approach we currently used in our system.

This approach can be extended to a weighted averaging approach by keeping a weight for each original dimension and taking the weights into account when calculating the average. In this way more important dimensions can have a larger impact on the representative dimension.

4.2.2 Approach 2: Using One Original Dimension

Since all the original dimensions in a dimension cluster are similar, we could also use one of them to represent all the others. One way to do so is to randomly select an original dimension from the dimension cluster, such as selecting the first original dimension or the last original dimension in the cluster. A better way is to choose the cluster center of the dimension cluster since its dissimilarity to every other dimension in the cluster is less than or equal to the radius of the dimension cluster. If the cluster center happens to be a non-leaf node, then select its cluster center recursively until reaching an original dimension. If users know all the pair-wise relations, they can choose the one whose sum of correlations with all others is minimal.

4.2.3 Approach 3: Applying Principal Component Analysis

This approach applies principal component analysis or dimension scaling to the original dimensions composing the dimension cluster and uses the first principal component as the representative dimension. This approach has the advantage that it maintains the main characteristics of the original dimensions. However, it has the inherent disadvantages of principal component analysis, such as the generated dimensions have no intuitive meaning. Even so, it is an interesting approach that we plan to implement in the future.

The above approaches of generating representative dimensions we proposed are only several among many possible approaches. Selecting a most suitable approach is an issue that is related closely to the particular application domain. Due to this reason, we keep our system open to user customized representative dimension generating approaches. In Section 5, we will describe how to interactively explore the hierarchical dimension cluster tree.

Chapter 5

Interactive Dimension Hierarchy Visualization

5.1 Overview

The purpose of interactive dimension hierarchy visualization to allow users to interactively navigate and modify the hierarchical dimension cluster tree built by dimension clustering (see Chapter 4), and allow users to select dimension clusters from the tree for convenient and efficient exploration.

We adopt a radial, space-filling hierarchy visualization technique named Sunburst by Stasko and Zhang [58] to visualization the hierarchical dimension cluster tree. In a Sunburst display, deeper nodes of the hierarchy are drawn further from the center; child nodes are drawn within the arc subtended by their parents; the angle occupied by a node is proportional to the number of leaf nodes under it.

Sunburst has some advantages over other tree drawing algorithms. First, as one of the space-filling techniques, it uses more implicit containment and geometry features to present a hierarchy than tree drawing algorithms which utilize edges between nodes to indicate parent-child structure[58]. Second, compared to treemap, another space-filling technique, radial space-filling techniques are better at conveying the hierarchy structure [58].

Our Sunburst is different from Stasko and Zhang's Sunburst in that:

- we use color to convey the hierarchical structure;
- we allow users to interactively modify the hierarchical dimension cluster tree directly through the Sunburst display by simple drag and drop operations;
- we provide interactive brushing to users so that they can select clusters of interest from the hierarchical dimension cluster tree directly through the Sunburst display;
- we propose a new distortion technique for Sunburst display in a more intuitive way;
- we implement drill-up/roll-down, zooming in/zooming out and panning operations for Sunburst display.

Figure 5.1 shows our dimension hierarchy dialog. The main canvas in the dialog is the Sunburst display for a hierarchical dimension cluster tree (we call it the Sunburst hierarchy). Under the main canvas there is a status bar that can show messages. On the upper right side there are four function buttons for switching among different canvas modes followed by two action buttons. On the lower right side there is a slider for the zooming operation. At the bottom of the dialog, there is an OK button that is used to close the dialog.

There are four different canvas modes that can be switched between by clicking the four function buttons:

- Normal - in this mode, distortion, modification and selection are forbidden, and the status bar will show the name of the node that the cursor is placed on;
- Distort - in this mode, users can distort the Sunburst display by clicking and dragging the mouse;

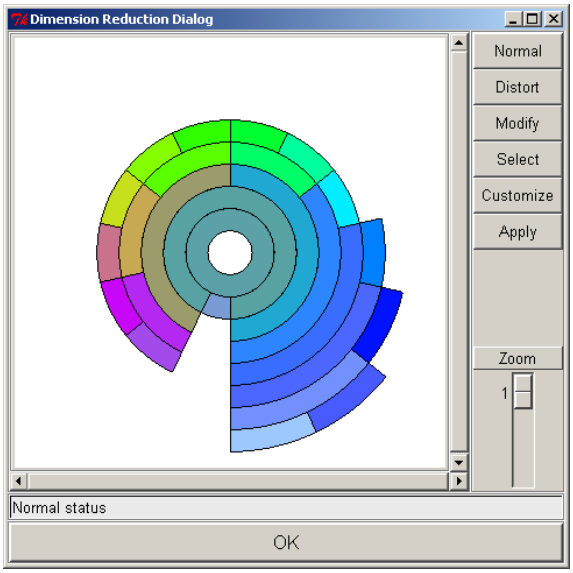

Figure 5.1: Dimension Reduction Dialog

- Modify - in this mode, users can modify the Sunburst hierarchy by clicking and dragging the mouse;
- Select - in this mode, users can highlight some nodes of the Sunburst hierarchy;

Details of distortion, modification, and selection are introduced in Sections 5.5, 5.3 and 5.4.

The two action buttons are used to:

- Customize - cause a customize dialog to pop up. Users can select among different degrees-of-dissimilarity representation methods from this dialog (see Chapter 6 for detail);
- Apply - use the highlighted nodes to construct a subspace in the multi-dimensional displays. (see Chapter 6 for detail).

In the following sections, we will discuss color assignment, modification, selection (brushing), distortion, and other interactive operations we implement for our Sunburst.

5.2 Color Assignment

In our Sunburst, we use the color of the nodes to convey the hierarchical structure of the dimension cluster tree. The principles of color assignment are:

- nodes belonging to the same cluster should have similar colors;
- a parent node's color should be a combination of the colors of its children;
- a larger child contributes more to its parent's color than its smaller siblings;

We have tried two different approaches to assign colors to the nodes of the sunburst: "middle color assignment" and "average color assignment". In both of these approaches, a whole linear colormap is mapped to the 360 degrees of the circle of the Sunburst display. We use hue as the variable for the colormap. In both approaches all the leaf nodes are assigned colors based on the degree of the center of their subtended angles. The difference lies in that in the middle color assignment, non-leaf nodes are assigned colors based on their position in the circle, while in the average color assignment, non-leaf nodes' colors are the average of their children's colors. The entry (the number of leaf nodes included in a cluster) of each child is used as a weight in the average calculation.

Figure 5.2 depicts the algorithm to assign color to a node using the middle color assignment approach, whereas Figure 5.3 shows the algorithm to assign color to a node using the average color assignment approach.

```
color MiddleColorAssignment(Colormap cm, DimensionCluster dc)
Begin
    double mid_angle =GetMediumAngle(dc.start_angle, dc.end_angle);
    return cm.MapColor(mid_angle);
End
```

Figure 5.2: Algorithm of Middle Color Assignment

```
color AverageColorAssignment(Colormap cm, DimensionCluster dc)
Begin
    If ( dc is a leaf node )
        Begin
            double mid_angle =GetMediumAngle(dc.start_angle,
dc.end_angle);
            return cm.MapColor(mid_angle);
        End
    Else ( dc is a non-leaf node )
        Begin
            For Each child C_i Do
                C_i.color =AverageColorAssignment(cm, C_i);
            Int R = 0, G = 0, B = 0;
            Int entries = 0;
            For Each child C_i Do
                Begin
                    R+ = C_i.color.R * C_i.entries;
                    G+ = C_i.color.G * C_i.entries;
                    B+ = C_i.color.B * C_i.entries;
                    entries+ = entries;
                End
            color.R = R / entries;
            color.G = G / entries;
            color.B = B / entries;
        End
    Endif
End
```

Figure 5.3: Algorithm of Average Color Assignment

Figure 5.4 shows a Sunburst display using the middle color assignment, while Figure 5.5 shows a Sunburst display using the average color assignment. Comparing these two figures, we find that Figure 5.5 reflects the hierarchical structure better than Figure 5.4. The reason is that in the average color assignment, a non-leaf node's color has a closer relationship with its children's colors - it is a mixture of them. Thus the parent-children relationship is more explicitly represented than in the middle color assignment. It makes sense since it will be close to the colors of the children with large entries - if a child

node's entry is large, it will gain a large weight in the parent node's color calculation. In the middle color assignment, only if the color scale reflects a strict "linear" numeric scale, would the relationship between the parent's color and children's color be revealed clearly. However, it is very difficult to get such an ideal color scale. It seems that the average color assignment fits our requirement better. Thus we adopted this approach.

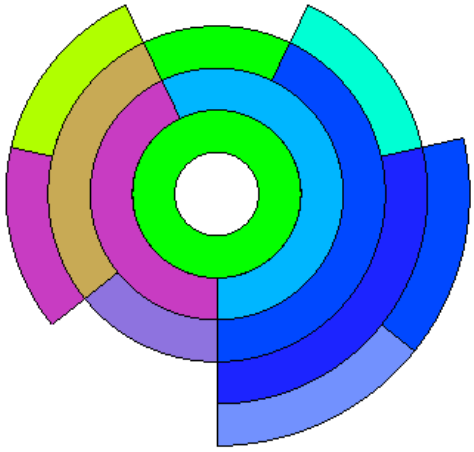

Figure 5.4: Dimension Hierarchy of Cars Data Set Colored Using Middle Color Assignment

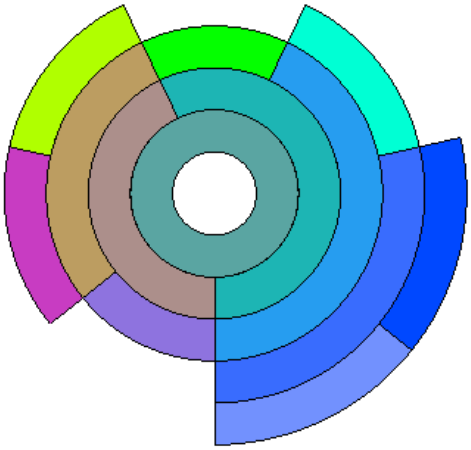

Figure 5.5: Dimension Hierarchy of Cars Data Set Colored Using Average Color Assignment

5.3 Modification Functions

We provide the following modification functions to users for two reasons:

- Although users can adjust the dimension clustering process by setting different parameters and changing similarity calculation methods, it is still an automatic process. Users cannot interactively take part in the dimension clustering process.
- Users could be experts with the data sets being visualized. They sometimes know that some relationships exist in the data sets according to their experience and

knowledge in their fields. These relationships could be undetected by the algorithm. Hence allowing users to interactively adjust the generated dimension cluster tree benefits the whole process.

Using our modification function, users can remove a sub-cluster from a cluster by dragging it and dropping it into any other cluster to be its new parent. The color of the removed cluster is retained to help users keep track of it. Figures 5.6 and 5.7 show a highlighted cluster being removed with all its children.

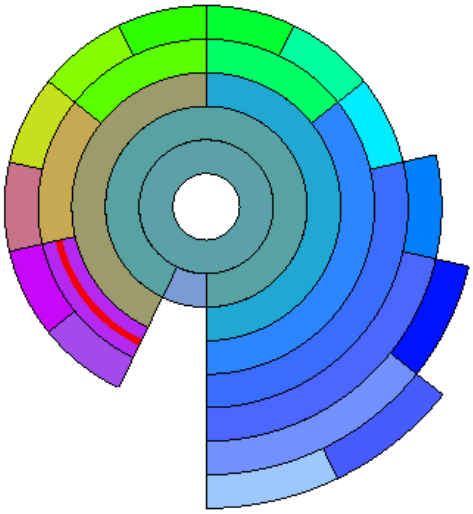

Figure 5.6: A cluster is highlighted.

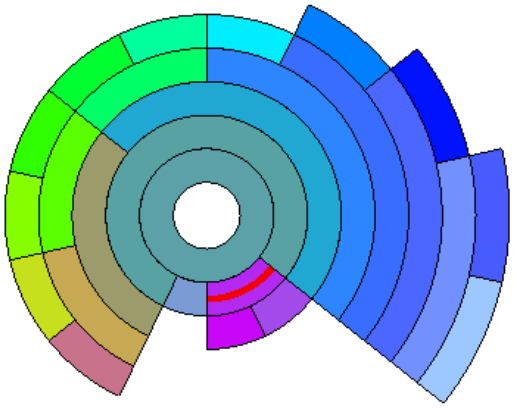

Figure 5.7: The highlighted cluster has been moved

In the future, we may explore use of highlighting or animation to help users perform the restructuring process more intuitively.

5.4 Brushing

The purpose of brushing in the context of our Sunburst display is to allow users to select dimension clusters efficiently for the sake of constructing a lower dimensional subspace for the data display. We have implemented two different brushing mechanisms. One is

called “simple brushing” and the other is called “structure-based brushing”.

Simple brushing is just a clicking operation. When users click the left button of the mouse on a cluster of the Sunburst display, this cluster will be highlighted and added to the end of the selected cluster list. When a user clicks it one more time, it will return to its normal color and be deleted from the selected cluster list.

Structure-based brushing allows users to select multiple clusters at the same time. When users click the right button of the mouse on a cluster of the Sunburst display, a small dialog pops upt. Users can select a threshold between 1 and the numbers of leaf nodes of the cluster through this dialog. We can consider the cluster and all its descendants as a tree rooted at the cluster. A structure-based selection will be done on this tree using the threshold. The selection starts from the root cluster. If its number of leaf nodes is less than or equal to the threshold, then it is selected and the process stops. Otherwise this process is repeated for all its children. Using this brush, we can select approximately uniformly sized clusters and cover all the leaf nodes of the root cluster at one time. We define a leaf node as covered if either itself or one of its ancestors is selected. When the threshold equals to 1, it means all the leaf nodes of the cluster will be selected. When the threshold equals to the number of leaf nodes of the cluster, then this cluster itself will be selected. Figure 5.9 shows the selection result of applying the structure-based brushing to the root node using a threshold of 3.

Since the structure-based brushing works on a cluster, we can apply it multiple times to different clusters using different thresholds. Hence we can select clusters in different levels of detail in different regions.

A variation of structure-based brushing is to allow users to input a threshold between 0 and the global degree of dissimilarity of the cluster. Then we select the sub-clusters that have global degrees of dissimilarity lower than or equal to this threshold. Thus the selected clusters would have approximately the same degree of dissimilarity.

In our data space, we highlight the selected data items by redrawing them using a highlight color. In the dimension hierarchy visualization, we want a similar approach to maintain the consistency of the system. We have tried two different highlighting strategies for selected clusters: "highlight all strategy" (see Figure 5.8) and "highlight part strategy" (see Figure 5.9). The first strategy is to cover the whole node with the highlight color in the Sunburst display, so we named it the "highlight all strategy". The original color of a node conveys information about the hierarchical structure. It no longer conveys such information when it becomes the highlight color. Thus our second strategy, named "highlight part strategy", is to highlight only part of the node. Hence the node is highlighted and the structure information is remained. We believe that the second strategy is better than the first.

For very large dimension hierarchies, adjacent leaf nodes will be assigned a very similar color. To overcome this problem, we can divide the 360 degree of the Sunburst display into several segments and map the whole color scale to each of the segments. Thus the color distance of adjacent leaf nodes would be enlarged.

We have provided users an option of showing names for selected clusters. Once this option is selected, the names of the selected clusters are shown on the display, with the names' left bottom corners located at the centers of selected clusters (see Figure 5.10.

5.5 Distortion

In the related work section, we mentioned several different distortion approaches to provide focus+context in Sunburst, such as Andrews and Heidegger's [56] two semi-circular approach and Stasko and Zhang's angular detail, detail outside, and detail inside approaches. Those approaches have the following disadvantages:

- There is a big visual "jump" before and after the distortion operation. That is, the

display before the distortion is quite different from the display after the distortion. This disadvantage is often remedied using complex animation so that users can follow the changes;

- They need more space for the focus + context display than the original display;
- It is difficult to have multiple levels of detail in one display.

To overcome these disadvantages, we propose a new distortion approach with the following features:

- It is easy for users to follow with simple animation;
- No extra space is needed for the focus + context display compared to no distortion;
- There can be multiple levels of detail in one display.

However, it should be noticed that after the distortion the angles occupied by the nodes are no longer proportional to their sizes.

The principle of this distortion approach is similar to the following valve moving process. Let's consider applying a distortion to a child of a parent cluster. In Sunburst, all the direct children of a parent cluster are in the same radius level of the display, inside the angle range of the parent cluster. We consider the left and right boundary of this angle range as a fixed left and right "baffle" limiting movement of its children. Also, we consider the radius level as a "channel". The first child is adjacent to the left baffle and the second child. The last child is adjacent to right baffle and the next to last child. All the other children are adjacent to their previous and next siblings. We view the boundaries between the children as "valves" that can slide along the channel. Two adjacent valves and the channel are the boundary of a child cluster. We can imagine that the clusters are filled with air.

When we want to enlarge a child cluster, we can fix its left valve, and move the right valve outwards. As a consequence, the air between the moved valve and the right baffle is compressed so that the valves in this area also move uniformly. Hence all the children after the enlarged child are uniformly compressed. We can also fix the right valve and move the left valve outwards. In this case all the children before it would be compressed uniformly.

When we want to deemphasize a child cluster, we can fix its left valve, and move the right valve inwards. As a consequence, the air between the moved valve and the right baffle is expanded so that the valves in this area also move uniformly. Hence all the children in the right side of the decreased child are uniformly enlarged. We can also fix the right valve and move the left valve inwards. In this case all the children before it would be enlarged uniformly.

Whenever a cluster is enlarged or decreased, the size of all its decendents also change proportionally so that they are always in the angle range of the cluster. A baffle of a cluster is a valve from the viewpoint of its parent cluster. So whenever we move the baffle of a cluster, the cluster as a whole is distorted.

Since this distortion is localized to a cluster, we can apply distortion to different clusters. We can even apply distortion to a cluster that is a child of another cluster that has already been distorted. In this way we can achieve multi-level distortion. Moreover, we don't need extra space for the distorted view.

When we implement this distortion operation, we have to make some changes to the ideal situation:

- We set a minimum angle for the leaf clusters. Hence the minimum angle of a cluster is its number of leaf nodes times the minimum angle of leaf clusters. A cluster cannot be compressed any more if it reaches its minimum angle. At this moment, the air in this cluster cannot be compressed any more.

- Users distort the display by dragging a valve. However, a valve between two clusters can be understood as the right valve of the left cluster or the left valve of the right cluster and the distortion consequence could be different. Thus we stipulate that if the cursor is closer the left cluster, we view the valve as the right valve of the left cluster, otherwise it will be viewed as the left valve of the right cluster.

Figure 5.11 shows a distortion process. In the left Sunburst, cluster 2 is to be decreased in size. Its right valve is fixed and its left valve is moved inwards. Cluster 1 is its only sibling that is before it. So it will occupy all the angle cluster 2 gives up. The figure on the right is the result of this distortion.

5.6 Other Interactive Operations

We have implemented many other interactive operations besides distortion in the Sunburst display, such as drilling-up/rolling-down, zooming in/zooming out, panning, and rotating operations.

Drilling-up/rolling-down are used to hide/show all the descendants of a cluster by simple mouse clicking. It helps users to prevent the display of some branches that they don't want to study. Figures 5.12 and 5.13 show a Sunburst display before and after hiding a large branch.

Zooming in/zooming out and panning operations allow users to enlarge the canvas and move around to examine details of the display. Figure 5.14 shows the same Sunburst of Figure 5.13 after zooming in and panning.

The rotating operation is a unique and necessary operation for the circular Sunburst display. In the distort mode, users can rotate the Sunburst display around its center in both directions by clicking mouse buttons in the blank place in the Sunburst display window. This operation helps users rotate their interested clusters to their favorite angles and avoid

the clutter of the names of the selected clusters. Figure 5.14 shows the same Sunburst of Figure 5.13 after rotating it approximately 100 degrees anti-clockwise.

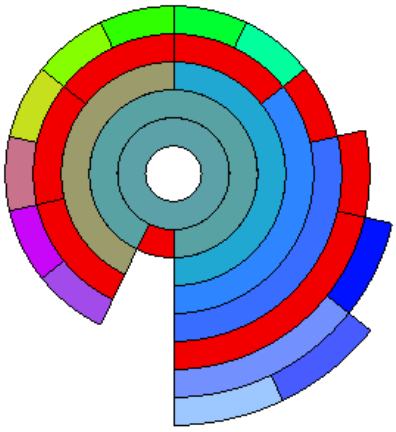

Figure 5.8: Highlight All Strategy

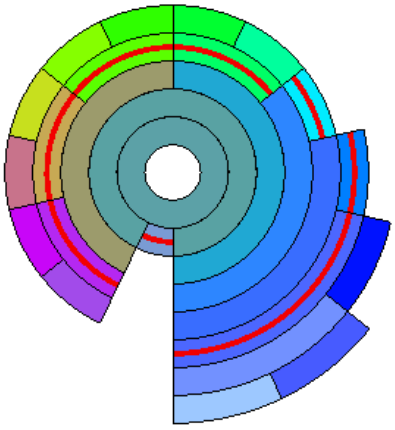

Figure 5.9: Highlight Part Strategy. Highlighted clusters are selected by applying the structure-based brushing to the root node. The threshold is 3.

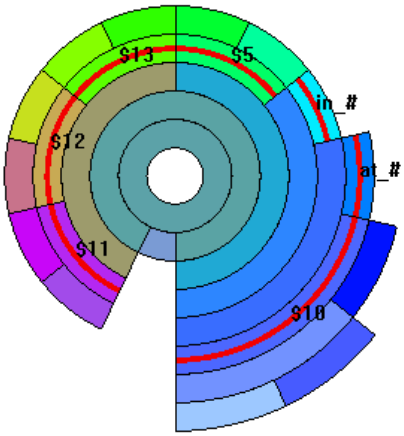

Figure 5.10: The names of the highlighted clusters are shown.

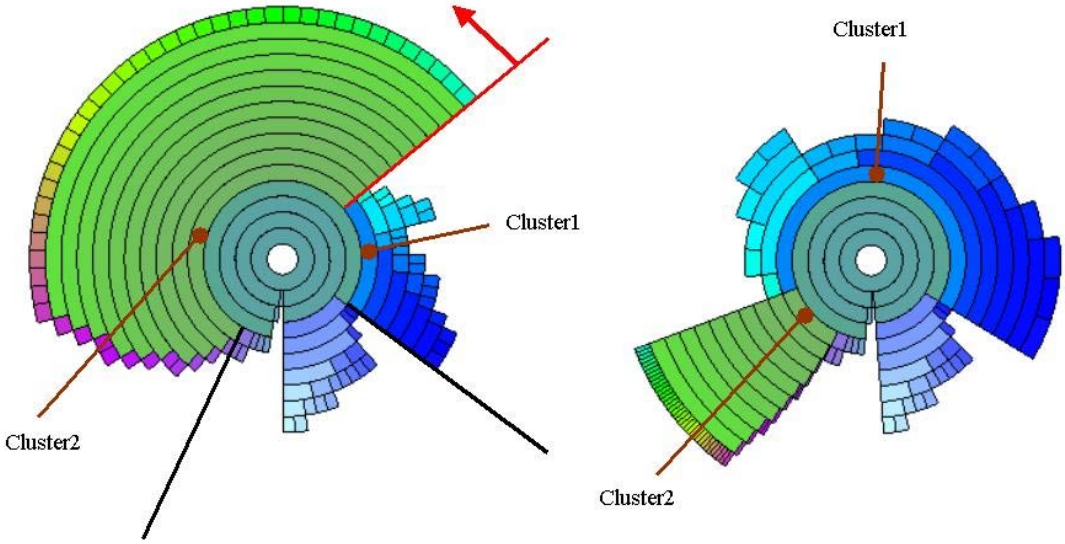

Figure 5.11: Distortion in Sunburst

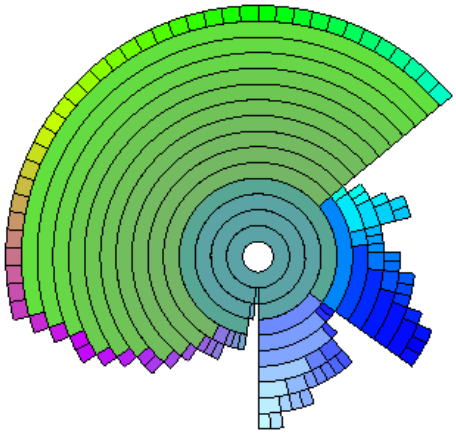

Figure 5.12: A Sunburst Display

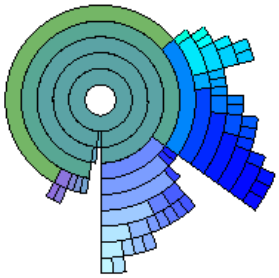

Figure 5.13: A big branch of the Sunburst in Figure 5.13 has been hidden.

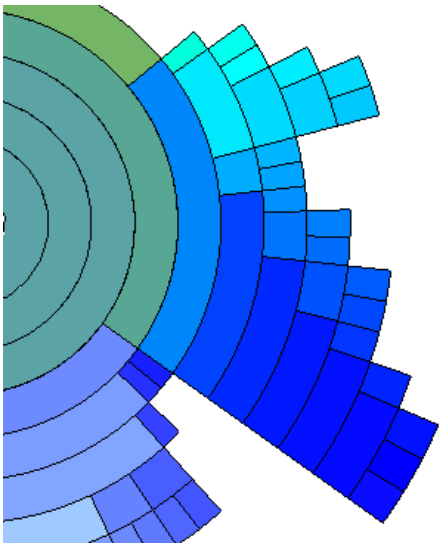

Figure 5.14: Sunburst in Figure 5.13 has been zoomed in and panned to view the detail of its middle right half.

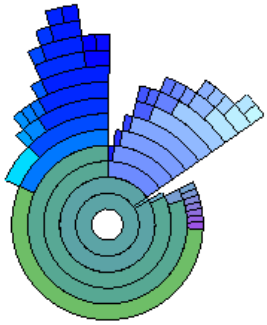

Figure 5.15: Sunburst in Figure 5.13 has been rotated approximately 100 degrees anti-clockwise.

Chapter 6

Applying Visual Dimension Reduction to Multidimensional Visualization

6.1 Overview

The ultimate goal of our approach is to reduce the dimensions displayed in the multi-dimensional visualizations to decrease the clutter problem and to increase the drawing speed. After the second step (described in Chapter 5), we have selected a set of representative dimensions (the number of the representative dimensions is much less than the number of original dimensions in the data set). Now we need to apply this reduction to the existing multi-dimensional visualizations, that is, to map the original data set from the original high dimensional space to a lower dimensional subspace. The result of this mapping can be visualized using existing multidimensional visualization techniques.

In this step, we have to address the following problems:

1. How to implement this mapping with low expense in space and time?

2. How to preserve useful information in the visualizations?

We address the first problem in Section 6.2, and then discuss our solutions to the second problem in Section 6.4.

6.2 Mapping

There is an ”Apply” button in the Sunburst dialog. When users click this button, our system will collect the pointers to the currently highlighted dimension clusters into a selected cluster list. Given a data item in the original data set, we can calculate its image in the lower dimension space composed of the representative dimensions of the selected clusters according to this list. Suppose there are N selected clusters in the selected cluster list, and data items in the original data set are stored in a double array old_data, we get the algorithm showed in Figure 6.1:

```
double* MapData(double* old_data, int N)
Begin
   double new_data = newdouble[N];
   For ( int i = 0; i < N; i++ )
   Begin
      double value =getRepresentativeDimensionValue(i, old_data);
      new_data[i] = value;
   End
   Return new_data
End
```

Figure 6.1: Algorithm of Data Mapping

There are two options for mapping the whole data set from the original high dimensional space to the lower dimensional space:

- Option 1: After every time the selected cluster list is updated, map all the data items of the original data set, one by one, and store their images into some data structure in memory. When redrawing the multi-dimensional displays, directly read from this data structure as the input to the multi-dimensional displays. We call this approach “pre-mapping”;

- Option 2: For every redrawing of the multi-dimensional displays, for every data item, we read it from the original data set, map it to the lower dimensional space

according to the selected dimension cluster list, and then draw the mapping result. We call this approach “online-mapping”.

The pre-mapping approach seems to be more efficient since it only does the mapping once for every selected cluster list update, no matter how many times we redraw the displays. However, it requires memory to store the mapping results. Memory is a critical resource when displaying large data sets, since the original data set already occupies a large amount of memory. Moreover, when users update the selected cluster list often (which can happen when users are in search of the best lower dimensional space), this method’s advantage in time savings will disappear.

The online-mapping approach needs no extra memory. But on a first glance, it wastes lots of time to recalculate the mapping result. However, compared to the time used to draw a data item on the screen, the time needed to calculate a mapping of that data item is negligible. So the online-mapping doesn’t have any significant difference for the response time to users. Thus the online-mapping is a good selection for us. We adopted this approach in our system.

In our system, all multi-dimensional displays share a selected dimension cluster list and a mapping function. So no matter to which display we apply the dimensional reduction, the other displays will be modified using the same mapping automatically.

6.3 Visualizations and Interactions

6.3.1 Visualizations

Generally speaking, applying the dimension reduction result to the existing multi-dimensional visualizations is simply using the mapped image of the data items as the input of the visualizations. Nothing is special here. However, we have applied the following techniques

to enhance the existing multi-dimensional visualizations to convey useful information to the user after the dimension reduction:

- We tell users how many original dimensions a cluster dimension contains by adding a number in parentheses after the name of the dimension. For example, in Figure 6.3, we can see that the first cluster dimension represents 48 original dimensions, while the second one represents 2 original dimensions.

- We provide the degree of dissimilarity within a cluster by various methods. We have tried several different approaches to conveying this dissimilarity information for clusters. Since the disagreement representation is an important topic that we have studied a lot, we will discuss it in detail in a separate section (Section 6.4).

Here is an example of the dimension reduction applied to the hierarchical parallel coordinates. Figure 6.2 shows the Ticdata2000 data set, which contains 86 dimensions and 5822 data items. Figure 6.4 shows its hierarchical dimension cluster tree automatically generated by our system in Sunburst. There are 9 clusters highlighted (marked by red arcs). These 9 clusters have been included in the selected dimension cluster list. Figure 6.3 shows the same data set mapped in the lower dimensional space composed of the representative dimensions of the selected clusters. Comparing Figure 6.2 and Figure 6.3, we find that the former is so cluttered that it is nearly impossible to interpret, while the latter is much clearer.

6.3.2 Interactions

All the existing interactive tools can still be used in the reduced dimensional displays. Besides those previously described, we generated a roll-up/drill-down interactive tool specifically for the reduced dimensional displays. This tool allows users to roll-up/drill-down the dimension clusters in the reduced dimensional displays by mouse clicking on

the axes without going to the Sunburst hierarchical dimension cluster tree dialog. The highlighted clusters in the Sunburst dialog will change automatically consistent with the reduced dimensional displays.

Figures 6.5 and 6.6 show an example of the drill-down operation. By right mouse clicking on the right-most axis in Figure 6.5, it has been drilled down to its two childen in Figure 6.6.

6.4 Degree of Dissimilarity (DOD) Representation

6.4.1 DOD Definition

Before we study the different methods for dissimilarity representation, we need to define the term "degree of dissimilarity"(DOD). We observe that two kinds of DOD exist, namely DOD for a single data item in every dimension cluster, and DOD for the data set as a whole in every dimension cluster. We name the former the "local degree of dissimilarity (LDOD)" and the latter the "global degree of dissimilarity (GDOD)". They are defined as follows:

- GDOD - the degree of dissimilarity for the entire data set in a dimension cluster. It is a scalar value. It is calculated according to the following rules:
  - For a leaf node that contains only one original dimension, its GDOD is zero.
  - Whenever a child is inserted into a cluster, we label the dissimilarity (see Chapter 4) between the representative dimension of the existing children in the cluster and the new child as "dis12", denote the old GDOD of this cluster as "dis1", mark the GDOD of the new child as "dis2". The new GDOD of this cluster is assigned as maximum$\{dis1, dis2, dis12 + dis1/2 + dis2/2\}$.

- LDOD - the degree of dissimilarity for a data item in a dimension cluster. It is represented by a mean, a maximum, and a minimum value. The mean is the mapped image of the data item on the representative dimension. The minimum is the minimum value among the values of the data item on all the original dimensions belonging to the dimension cluster. The maximum is the maximum values among the values of the data item on all the original dimensions belonging to the dimension cluster. Here all the dimensions have been normalized so values lie between 0 to 1.

We discuss our different approaches to graphically depicting LDOD and GDOD in the following subsections.

6.4.2 Approach 1: Mean-Band Method to Represent LDOD

The first method is an extension of the mean-band method in the interactive hierarchical displays (IHD) framework we built for the hierarchical displays [30]. It is used to represent LDOD here. In this approach, every data item in the reduced dimension displays is composed of a mean and a band. For each cluster dimension, the mean corresponds to the mean of the data item in that dimension cluster, that is, the mapped image of the data item on this cluster dimension, while the band ranges from its minimum to its maximum.

The mean-band method in the IHD framework was applicable to multiple multi-dimensional visualization techniques, as is the mean-band method here. We have successfully applied it to the flat parallel coordinates (see Figure 6.7). We are also able to apply it the flat star glyphs, scatterplot matrices, and dimensional stacking.

However, there are some drawbacks to this method:

- this method causes serious overlaps in the displays. The situation can be relieved by applying extent scaling to it, that is, proportionally reducing the bands of all the data items.

- this method increases the response time. Drawing a band for each data item in the display is time consuming.

- this method is difficult to be applied to the hierarchical displays since bands would then have be assigned another meaning in the hierarchical displays [29, 71, 72, 30].

Figure 6.7 shows the Detroit data set (7 dimensions, 13 data items). The second and fourth axis representing clusters containing 4 and 2 original dimensions respectively. The bands have been decreased by 70% percent. From the figure we find that most data items have larger LDODs in the former cluster than the latter.

Figure 6.8 shows the Aaup data set (14 dimensions, 1161 data items). The second and fourth axis represent non-leaf dimension clusters of size 6 and 5 respectively. Although the bands have been decreased by 85% percent, the figure is still seriously cluttered.

It seems surprising that similar principles have different effects (the mean-band method for disagreement representation and for hierarchical displays.) In fact, it is not so surprising after a careful analysis. In the hierarchical display, lots of data items are represented by one band, but for representing a single disagreement, nothing can be removed from the screen, while a band is added for every data item! We hence conclude that the mean-band method is not suitable for application to this problem except for cases with very small numbers of data records.

6.4.3 Approach 2: Three-Axes Method to Represent LDOD

The basic idea of this method is to use two extra axes around a cluster dimension to indicate the minimum and maximum of the dimension cluster for every data item. These extra axes can be viewed as common dimensions in the visualizations. The three-axes method can be applied to the flat and hierarchical parallel coordinates and star glyphs.

In the flat parallel coordinates, two extra axes are arranged closely on each side of a

cluster dimension axis (middle axis) that represents more than one original dimension. The minimums of the dimension cluster of the data items are mapped to the left side axis, and the maximums are mapped to the right side axis. These three axes together form a group that can be distinguished from other dimensions. Moreover, the left side, middle and right side axes are assigned different colors to help users to distinguish among them. In the hierarchical parallel coordinates, the axes are arranged in the same way as in the flat parallel coordinates. The only difference is that the means and bands representing clusters instead of single lines representing single data items are mapped to these axes.

Figures 6.9 and 6.10 show the Aaup data set (14 dimensions, 1161 data items) in flat and hierarchical parallel coordinates. The second and fourth red axis represent non-leaf dimension clusters of size 6 (dimension cluster 1) and 5 (dimension cluster 2) respectively. The dark yellow and dark blue axes adjacent to them are their left side and right side axes. In Figure 6.9 we can see that the highlighted data items have larger LDOD in dimension cluster 2 than in dimension cluster 1, i.e., their projections on the right side axis are much higher than the left axis for cluster 2. From Figure 6.10 using IHD, we find that the yellow data cluster has a LDOD in dimension cluster 1 than the blue data clusters, while in dimension cluster 2, it has a smaller LDOD than the blue data cluster, although it is larger in size.

The three-axes method has the advantage that it works on the flat and hierarchical displays equally well, since it doesn't change the nature of the displays. It works much faster than the mean-band method, while allowing users to see the LDOD with less overlap. However, it adds extra axes to the display. We feel it is not suitable for the scatterplot matrices since it will increase the number of the plots significantly. Also it is not suitable for the star glyphs, since in the star glyphs, each star glyph only occupies a small space, so extra arms clutter the display.

6.4.4 Approach 3: Using Diagonal Plots to Represent LDOD in Scatterplot Matrices

We have implemented a unique way to represent LDOD in the flat scatterplot matrices. We noticed the fact that the diagonal plots in the scatterplot matrices convey little useful information; all the points clutter in the diagonal lines in these plots. Thus we designed a new way to use this space of these plots to convey LDOD.

In the flat scatterplot matrices, the x and y coordinates of a diagonal plot are the same dimension. In our reduced dimensional space, if the dimension is a representative dimension, we map the minimum and maximum of the dimension cluster to the x and y coordinates of its diagonal plot. Thus the display of the diagonal plot would no longer contain points cluttering in the diagonal lines if the LDODs of this dimension cluster are non-zero. Rather the points are now spread out in the plot. A point far away from the diagonal line means a large LDOD.

Figures 6.11 and 6.12 show the Aaup data set (14 dimensions, 1161 data items). The second and fourth axes represent non-leaf dimension clusters of size 6 (cluster 1) and 5 (cluster 2) respectively. Figure 6.12 used the diagonal plots (special colors are used on their boundary to distinguish them from other plots) to represent the LDODs of the data items, while Figure 6.11 does not. Comparing the two figures, we find Figure 6.12 does not introduce any extra overlaps. Moreover, it indicates that cluster 2 has a large degree of disagreement since in the fourth diagonal plot, the points are scattered in the plot, with some points far away from the diagonal line. Although cluster 1 has a larger size than cluster 2, it has smaller degrees of disagreement for most data items. However, there is a small portion of data items that do have significant LDODs that form a small area far away from the diagonal line.

This method puts forward a new way to make use of the diagonal plots in the scatter-

plot matrices and does not introduce extra overlaps. By this method, users can extract the disagreement extent of a representative dimension from the display. This is the best way we have discovered thus far for conveying disagreement information for the flat scatterplot matrices.

6.4.5 Approach 4: Outer and Inner Stick Method to Represent LDODs in Star Glyph

The outer and inner stick method adheres some small sticks to the star glyphs to convey LDOD. An outer stick has a length of $(maximum - mean)$. One of its ends is placed at the end of an arm with the other end extending outward along the directions of the ray. An inner stick has a length of $mean - minimum$. One of its ends is placed at the end of an arm with the other end extending inward along the direction of the ray. The arm, the outer stick and the inner stick have distinguishable colors so that users can easily perceive them. By watching the length of the inner sticks and the outer sticks, users can detect the LDOD of a data item (a glyph) easily.

Figures 6.13 and 6.14 show the same part of the Aaup data set (14 dimensions, 1161 data items) using the flat star glyph. The second arm (72 degree) and fourth axis (216 degree) represent non-leaf dimension clusters of size 7 (dimension cluster 3) and 6 (dimension cluster 1) respectively. Figure 6.14 uses the outer and inner stick method (outer sticks are blue and inner sticks are red) to represent the LDODs, while Figure 6.13 does not. Comparing the two figures, we find Figure 6.14 does not introduce any extra overlaps. Moreover, it indicates that most data items have large LDODs in dimension cluster 3 and small LDODs in dimension cluster 1. We also find an exception that the star glyph at the right bottom corner bears a large LDOD in dimension clusters that makes it different from other data items.

6.4.6 Approach 5: Axis Width Method to Represent GDOD

The idea of the axis width method is to use the width of a representative dimension axis to convey the GDOD for the dimension cluster it represents. The larger the disagreement, the wider the axis. It can be applied to various flat and hierarchical displays. Currently, we have applied it to the flat and hierarchical parallel coordinates, scatterplot matrices and star glyphs. The width of an axis is calculated using the following equation:

$$width = 1 + whole\_disagreement * 10$$

Figures 6.15 and 6.16 show the Aaup data set (14 dimensions, 1161 data items) in flat and hierarchical parallel coordinates with the wide axes method applied to show GDOD. The second and fourth red axes represent non-leaf dimension clusters of size 6 (dimension cluster 1) and 5 (dimension cluster 2) respectively. It can be seen that dimension cluster 2 has a larger GDOD than dimension cluster 1. This is consistent with the conclusion we draw by watching LDODs using other approaches. Figures 6.17 and 6.18 are the counterparts of Figure 6.15 and Figure 6.16 in flat and hierarchical scatterplot matrices.

The axis width method can also be applied to the star glyphs. However, it does not work well for the star glyphs. The reason is that since each star glyph only occupies a small space, wide axes (arms) clutter the displays. From another point of view, the wide axes method is an approach to represent GDOD, while there are no global axes in the star glyphs that show global properties.

6.4.7 Discussion

Some of these methods can be applied to multiple multi-dimensional visualizations, while some are only suitable for particular visualization techniques. All of them have their own advantages and disadvantages.

In future work, we want to try to develop one uniform approach suitable for most or

ideally for all displays. It should have the following features:

- it should be understandable by users;
- it should not introduce too much overlap; and
- it should not increase the response time significantly.

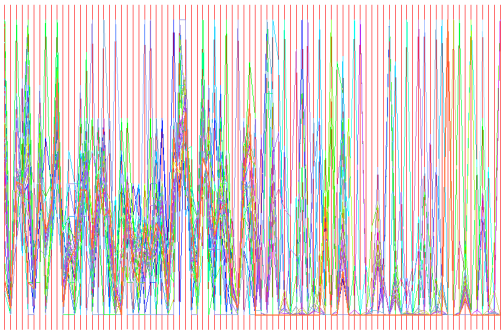

Figure 6.2: Ticdata2000 Data Set (86 dimensions, 5822 data items) in Hierarchical Parallel Coordinates

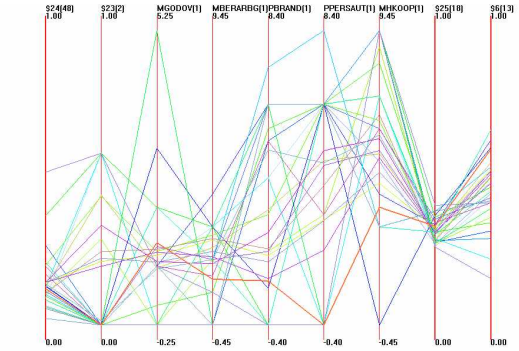

Figure 6.3: Ticdata2000 Data Set Mapped to a 9 Dimensional Subspace in Hierarchical Parallel Coordinates

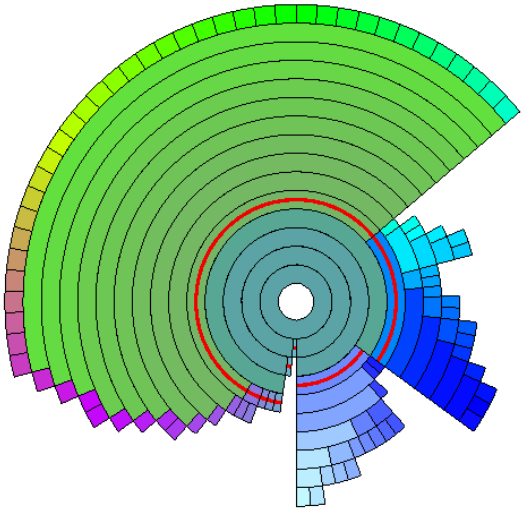

Figure 6.4: Hierarchical Dimension Cluster Tree of Ticdata2000 Data Set in Sunburst

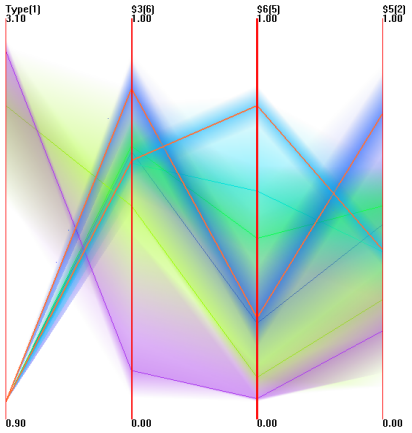

Figure 6.5: Aaup data set in Hierarchical Parallel Coordinates before drilling down the right-most axis.

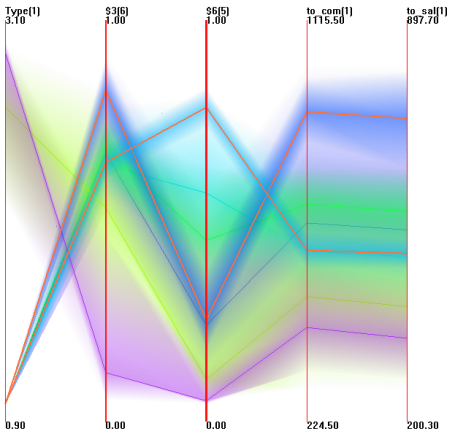

Figure 6.6: Aaup data set in Hierarchical Parallel Coordinates after the drill-down operation.

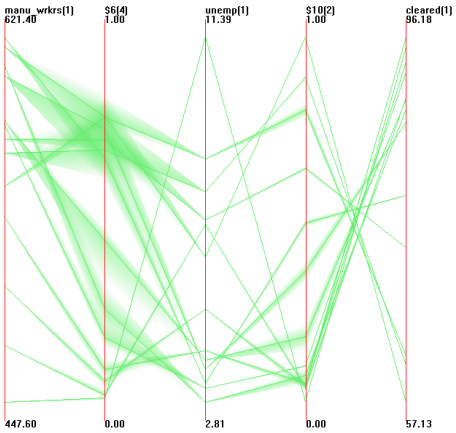

Figure 6.7: Detroit data set in Flat Parallel Coordinates. The second and fourth axis are non-leaf dimension clusters. The bands are decreased by 70%.

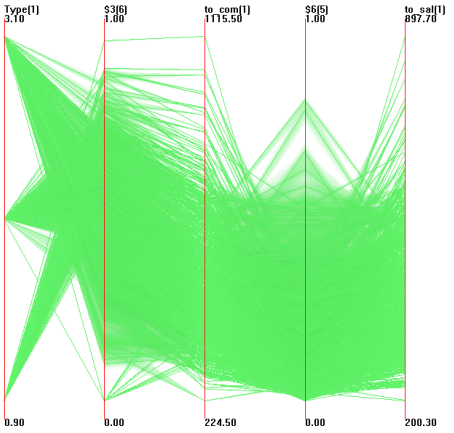

Figure 6.8: Aaup data set in Flat Parallel Coordinates. The second and fourth axis are non-leaf dimension clusters. The bands are decreased by 85%.

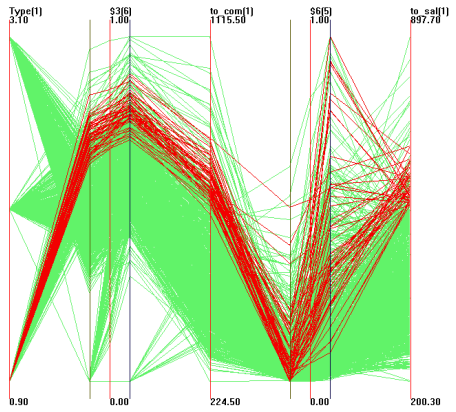

Figure 6.9: Aaup data set in Flat Parallel Coordinates. The second and fourth axis are non-leaf dimension clusters. Some data itemed are highlighted in red.

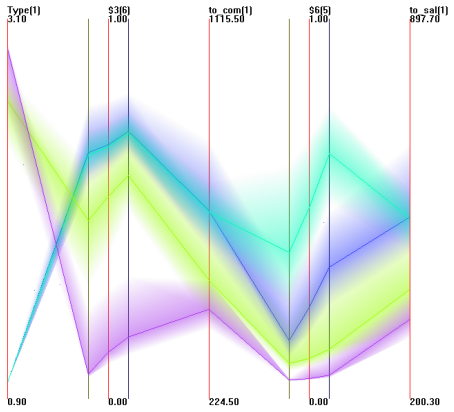

Figure 6.10: Aaup data set in Hierarchical Parallel Coordinates. The second and fourth axis are non-leaf dimension clusters. The data bands are decreased 60%.

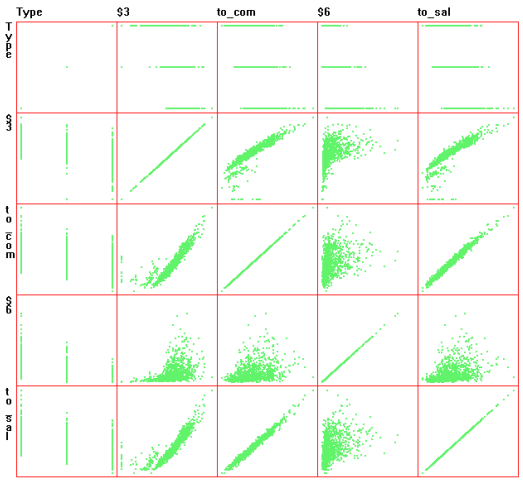

Figure 6.11: Aaup data set in Flat Scatterplot matrices. The second and fourth axis are non-leaf dimension clusters.

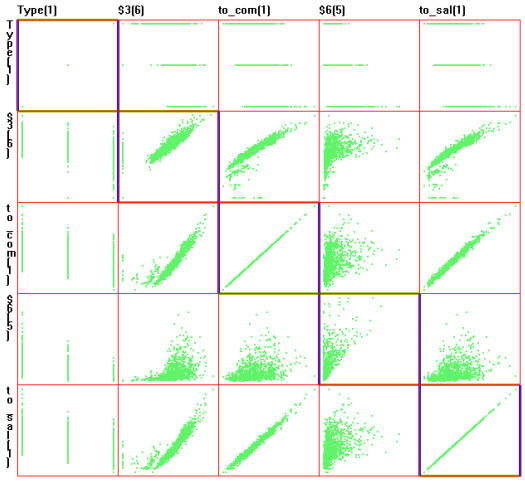

Figure 6.12: Aaup data set in Flat Scatterplot matrices. The second and fourth axis are non-leaf dimension clusters. The diagonal plots are used to represent LDODs.

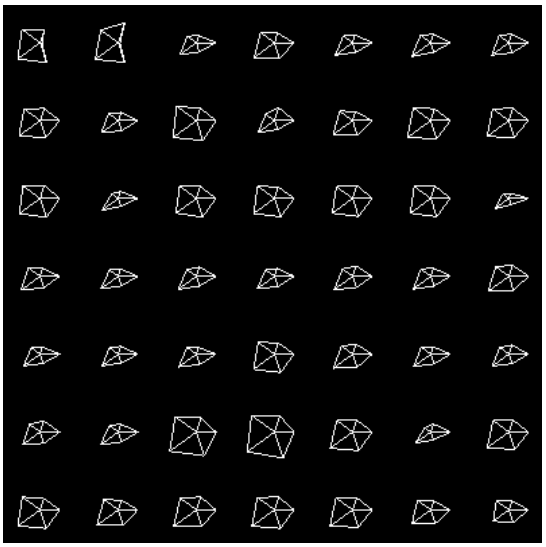

Figure 6.13: Part of Aaup data set in Flat Star Glyph. The arms at 72 degree and 216 degree represent non-leaf dimension clusters.

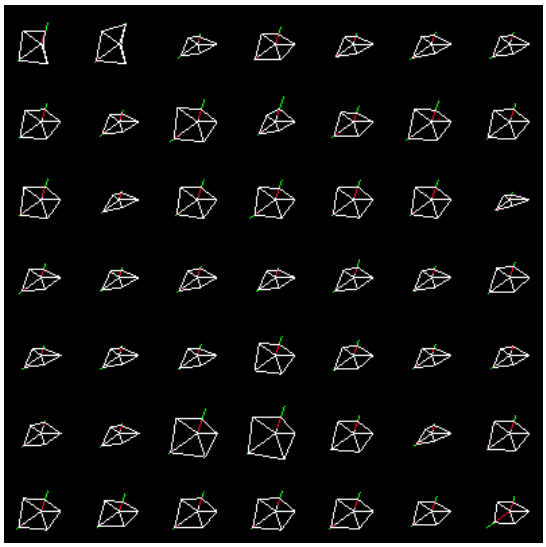

Figure 6.14: Part of Aaup data set in Flat Star Glyph. The arms at 72 degree and 216 degree represent non-leaf dimension clusters. The outer and inner stick method is applied to represent LDODs.

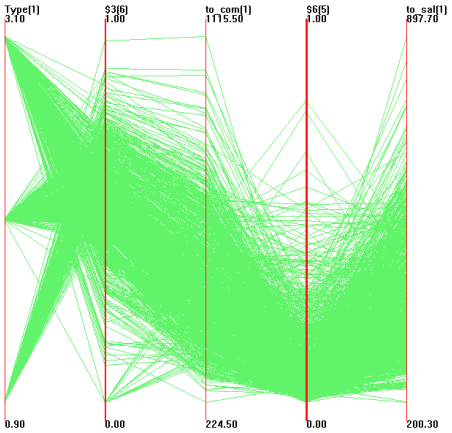

Figure 6.15: Aaup data set in Flat Parallel Coordinates. The second and fourth axes are non-leaf dimension clusters. The axis width method is applied to show GDOD.

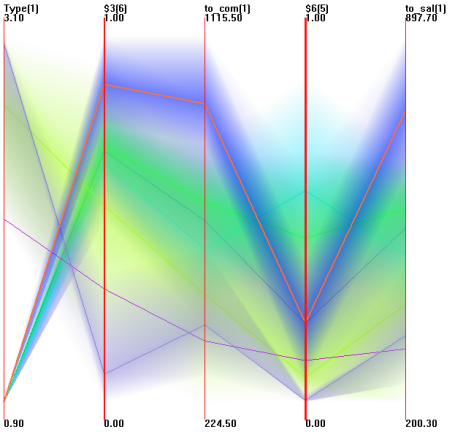

Figure 6.16: Aaup data set in Hierarchical Parallel Coordinates. The second and fourth axes are non-leaf dimension clusters. The axis width method is applied to show GDOD.

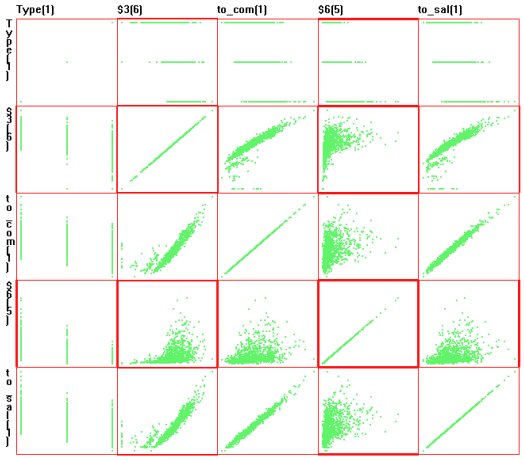

Figure 6.17: Aaup data set in Flat Scatterplot Matrices. The second and fourth axes are non-leaf dimension clusters. The axis width method is applied to show GDOD.

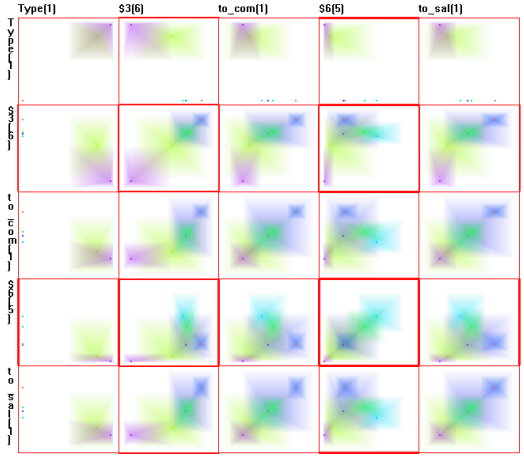

Figure 6.18: Aaup data set in Hierarchical Scatterplot Matrices. The second and fourth axes are non-leaf dimension clusters. The axis width method is applied to show GDOD.

Chapter 7

Implementation of the Visual Hierarchical Dimension Reduction

7.1 Overview

The proposed VHDR approach was implemented as extensions to the XmdvTool 4.2 system. Figure 7.1 shows a high-level diagram of how our extensions fit into the original XmdvTool system. The new modules introduced were:

- Dimension Cluster Module

  This module reads data items or data clusters from a data file or cluster file and generates a dimension hierarchy using a dimension clustering algorithm. It also assigns or generates a representative dimension for each dimension cluster in the dimension hierarchy.

- Dimension Hierarchy Display and Interaction Module

  This module visualizes the dimension hierarchy and enables interaction with the dimension hierarchy by users. Users can visually explore and modify the dimension

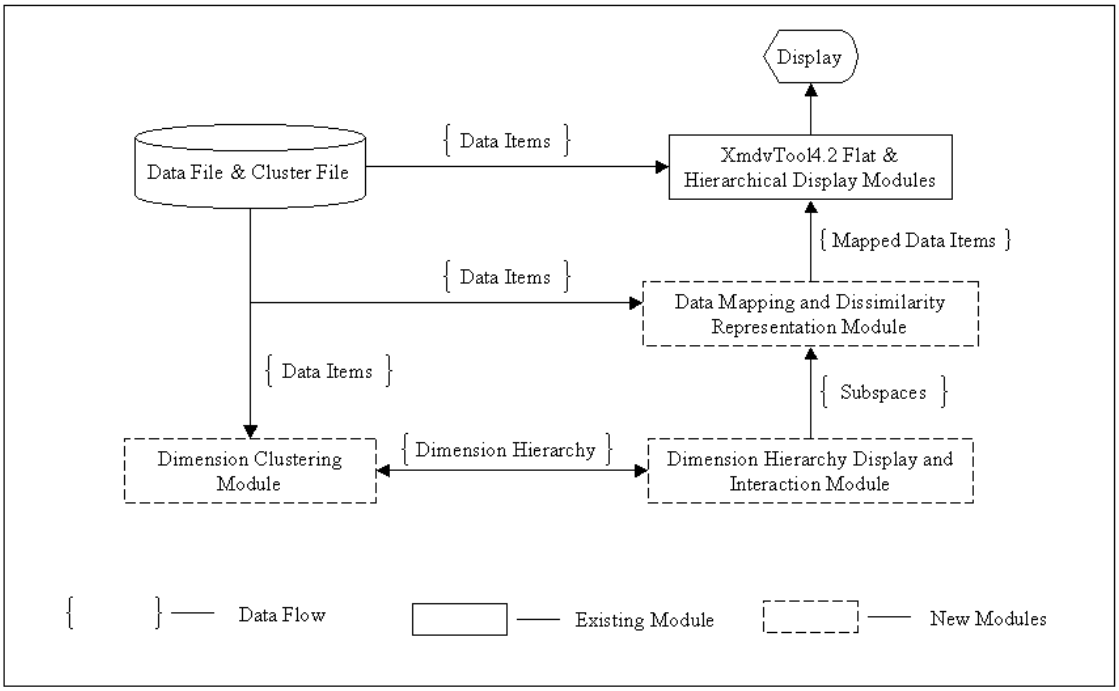

Figure 7.1: A high-level diagram of the modules in XmdvTool

hierarchy through this module. For dimension reduction, the users need to select some dimension clusters using brushing tools provided by this module.

- Mapping Data and Dissimilarity Representation Module

  This module maps data items and data clusters from original high dimension space to a lower dimensional subspace composed of the representative dimensions of the dimension clusters selected. It also provides dissimilarity representation. If the flat and hierarchical display modules use output from this module instead of the original data items and data clusters, then they generate views in the lower dimensional subspace.

7.1.1 Platform

This project is implemented as extensions to the XmdvTool system version 4.2. Similar to XmdvTool 4.2, it also implemented using C++ and the OpenGL graphics library. Its interface is generated using Tcl/Tk. This project can run on both Windows 95/98/NT/2000 and Unix platforms.

Chapter 8

Case Studies

We have conducted two performance case studies to illustrate the usefulness of our tool. In our tests we used a Census-Income data set, a 42 dimensional, 20,000 element data set derived from part of the unweighted PUMS census data from the Los Angeles and Long Beach areas for the years 1970, 1980, and 1990. Figures 8.1 and 8.2 show the hierarchical parallel coordinates and hierarchical scatterplot matrices display of the data set respectively. It is almost impossible to find any meaningful patterns from them without dimension reduction.

We use the algorithm described in Chapter 4 to generate the hierarchical dimension cluster tree. The percentage threshold used is 90%. The original data set is used in the clustering process. In Section 8.1, we describe how users can explore the hierarchical dimension cluster tree through flexible interactions with the Sunburst display. In Section 8.2, we illustrate how to find patterns from the data set in the reduced dimension subspaces.

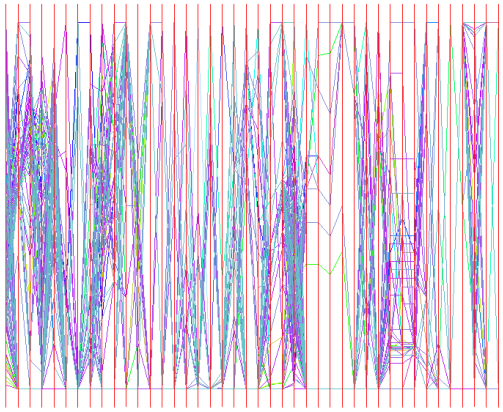

Figure 8.1: Census-Income Data Set (42 dimensions, 20,000 data items) in Hierarchical Parallel Coordinates

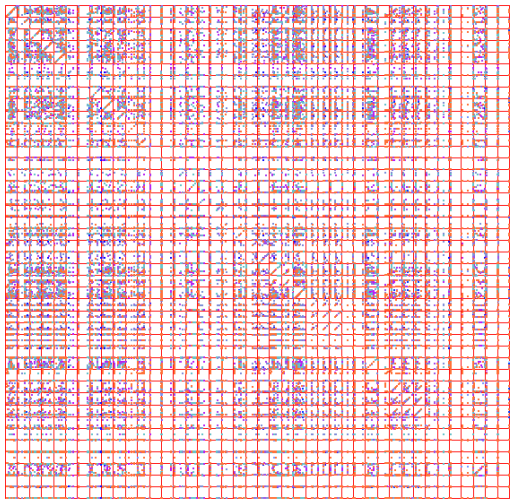

Figure 8.2: Census-Income Data Set (42 dimensions, 20,000 data items) in Hierarchical Scatterplot Matrices

8.1 Interaction in Sunburst

Figure 8.3 shows the hierarchical dimension cluster tree of the Census-Income data set in the Sunburst display. We interactively explored this tree by iteratively applying the following approaches:

- Using the structure-based brush to evaluate the degrees of dissimilarity of the dimension clusters in the tree, and to highlight the clusters we are interested in;
- Watching details of certain clusters that we are interested in using detail+context distortion, rotation, zooming in/zooming out, and panning operations;
- Modifying the tree if any parts of the tree looks unreasonable according to our experience.

Figure 8.4 shows the result of applying a structure-based brushing operation on the root node. The maximum dissimilarity allowed in this selection was set to 0.1. The selection result suggests that clusters $23, $52, $49 may be clusters with good correlation

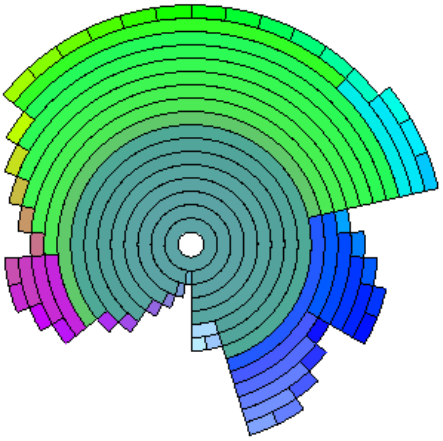

Figure 8.3: Hierarchical Dimension Cluster Tree of Census-Income Data Set

Figure 8.4: Struture-Based Brushing Applied to Root Node

of dimensions within them. Hence we un-selected all the clusters, and applied structure-based brushing to cluster $23 at threshold 0.0 to highlight its leaf nodes. Figure 8.5 shows all the leaf nodes of cluster $23. It reveals that the leaf dimensions of cluster $23 are "migration_code-change_in_msa", "migration_code-move_within_reg", "migration_code-change_in_reg", and "live_in_this_house_1_year_ago". By checking cluster $23 we believe that it should be a cluster with tight internal relationships according to our knowledge of the data sematitics. Similarly, we wanted to check cluster $52 in detail. However, we could not see the complete names of its leaf dimensions from the display since it was cut by the edge of the canvas (see Figure 8.6). Hence we rotated the display so that we can see the complete names (see Figure 8.7). From Figure 8.7 we found that cluster $52 is composed of "country_of_birth_mother", "country_of_birth_father", and "country_of_birth_self". It also seems reasonable to group these dimensions.

Figure 8.8 shows the details of cluster $49 after rotation. It seemed odd to us that some dimensions, such as "region_of_previous_residence" were put together with dimensions such as "income". According to our experience, we felt that they were not related.

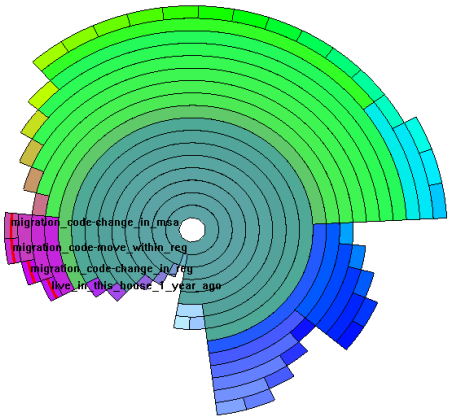

Figure 8.5: Details of Cluster $23

Hence we decided to remove some dimensions from cluster $49 and put them into the root cluster. Figure 8.9 shows the details of cluster $49 after the modification. In Figure 8.9, it is hard to see the details of the moved nodes since they are direct children of the root cluster, and thus they occupy a rather small display area and their names clutter together. Thus we used distortion, zooming in and panning operations and generated a focus+context view for the moved leaf nodes so that we now can view them in detail within the context (see Figure 8.10).

8.2 Applying Dimension Reduction to Multidimensional Displays

As a first experiment, we selected “education”, “age”, “sex”, “weeks_worked_in_year”, and “income” from clusters that had large dissimilarity among them. We viewed them as representatives of those clusters and hoped that they could form a subspace that could reveal the main trend in this data set.

After mapping the data set into the subspace composed by them, we have found some interesting patterns from the multi-dimensional displays. Figure 8.11 to Figure 8.14 are

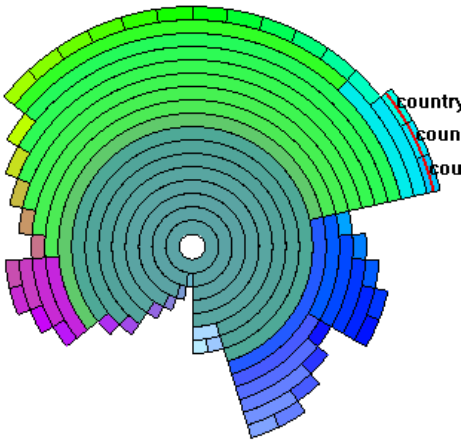

Figure 8.6: Before Rotation. Names of Highlighted Nodes are incomplete.

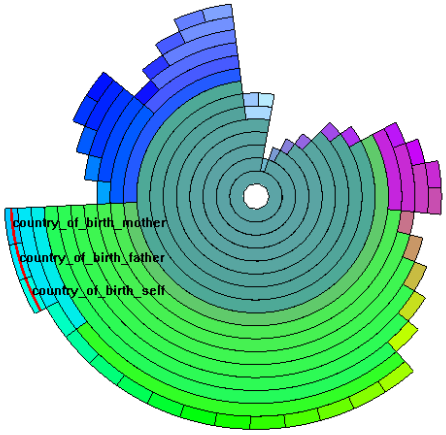

Figure 8.7: After Rotation. Names of Highlighted Nodes are complete.

hierarchical parallel coordinates in this subspace. The first axis (from left to right) in each figure is education (from under 1 grade (lowest) to Ph. D. (highest)). The second axis is age (from 0 years old (lowest) to 94 years old (highest)). The third axis is sex (woman: low, man: high). The fourth axis is weeks_worked_in_year (from 0 (lowest) to 52 (highest)). The last axis is income (less than 50,000 (low), equal to or more than 50,000 (high)).

Figure 8.11 shows a data cluster that can be interpreted as “a group of well-educated men who work most of the year and got high income”. Figure 8.12 shows the same cluster in a higher level of detail. As a counterpart, Figure 8.13 shows a data cluster that can be interpreted as “a group of not so well-educated men who did not work most of the year and got low income”. Figure 8.14 shows a data cluster that can be interpreted as “a group of not so well-educated women who did not work most of the year and got low income”.

As a second experiment, we wanted to verify that “country_of_birth_mother”, “country_of_birth_father”, and “country_of_birth_self” really have a close relationship among them. Thus we used these three dimensions to form a subspace and mapped the data set

into this subspace. Figure 8.15 reveals that they really have a close relationship. We can hardly find this pattern from Figure 8.15's counterpart view in the original 42 dimensional data space (see Figure 8.16).

As a third experiment, we explored the subspace consisting of the dimensions composing the modified cluster $49. We found two patterns from it; most low income people have low wage per hour and low capital gain (see Figure 8.17) and there are a few people of high income who have low wage per hour but have high dividends from stocks (see Figure 8.18).

High dimensional data sets, such as the Census-Income data set in the case studies, can reveal a wealth of information. It just requires the appropriate tools and the perceptual abilities of the user. Ours is one such tool.

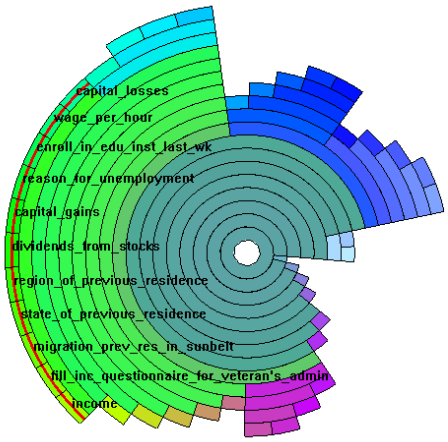

Figure 8.8: Some dimensions should not be in Cluster $49 according to our intuition.

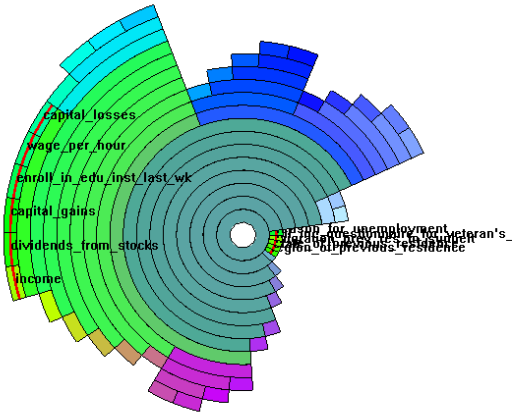

Figure 8.9: We moved these to become children of the root node. We cannot see the details of the moved out nodes in this view.

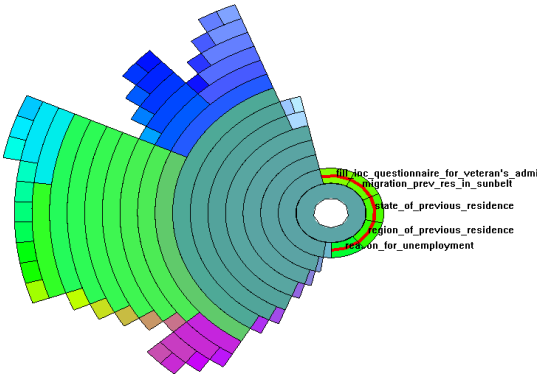

Figure 8.10: After distortion, we can see the details of the moved nodes.

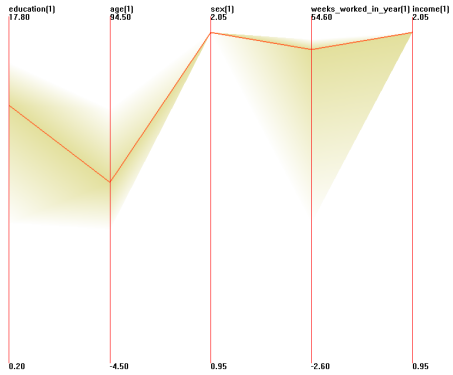

Figure 8.11: Pattern 1: a group of well-educated men who work most of the year and got high income.

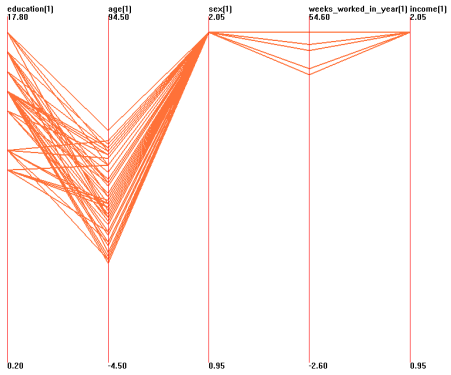

Figure 8.12: Pattern 1 in more detail

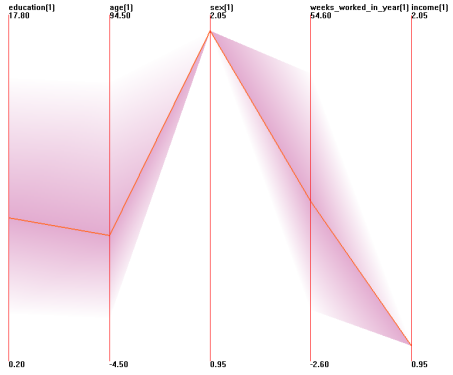

Figure 8.13: Pattern 2: a group of not so well-educated men who did not work most of the year and got low income.

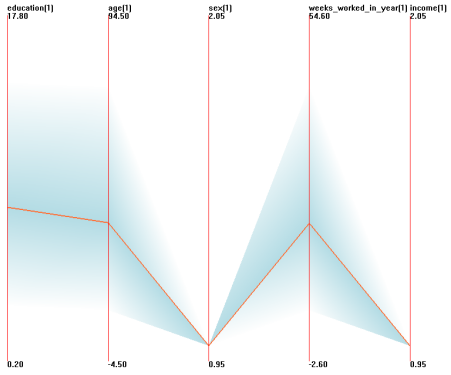

Figure 8.14: Pattern 3: a group of not so well-educated women who did not work most of the year and got low income.

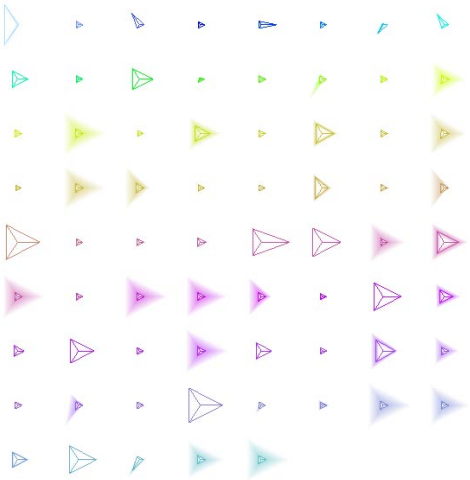

Figure 8.15: Hierarchical Star Glyphs in Subspace. country_of_birth_mother: 0 degree, country_of_birth_father: 120 degree, country_of_birth_self: 240 degree. Arms of the same length mean the same country.

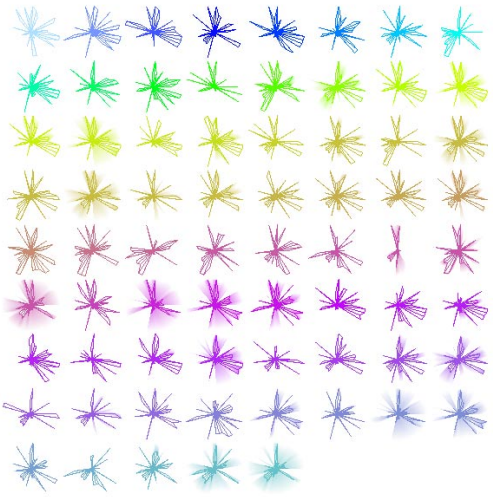

Figure 8.16: Hierarchical Star Glyphs in Original Space. country_of_birth_mother: 282.9 degree, country_of_birth_father: 274.3 degree, country_of_birth_self: 291.4 degree. Arms of the same length mean the same country.

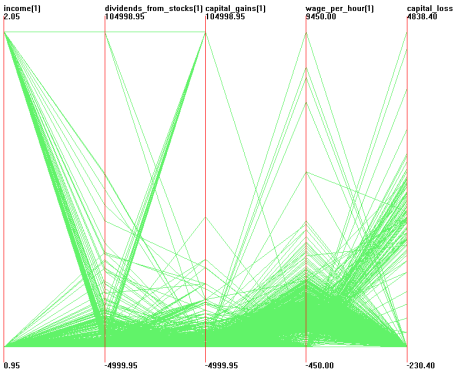

Figure 8.17: Flat Parallel Coordinates in Subspace. The axes from left to right are: income, dividends_from_stocks, capital_gains, wage_per_hour, and capital_losses. The high and low in the axes is corresponding to the real number. Pattern 4: most people of low income have low wage per hour and low capital gain.

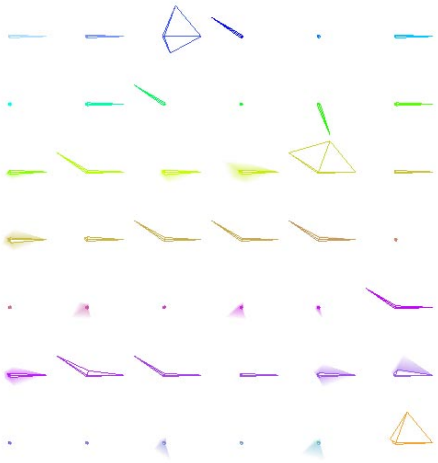

Figure 8.18: Hierarchical Star Glyphs in Subspace. income: 0 degree, dividends_from_stocks: 72 degree, capital_gains: 144 degree, wage_per_hour: 216 degree, capital_losses: 288 degree. A short arm means a small number and a long arm means a large number. Pattern 5: a few people of high income have low wage per hour but they got high dividends from stocks (the last glyph).

Chapter 9

Conclusions and Open Questions

9.1 Summary and Contributions

The issue of high dimensional data sets has implications in many research areas, such as data warehousing, multimedia, document visualization, survey analysis and so on. In multidimensional visualization, data sets with huge dimensionality result in display clutter. In this thesis, we have proposed a visual hierarchical dimension reduction approach and applied it to several popular multidimensional visualizations to present data sets with huge dimensionalities. Dimension reduction is not new. However, the existing dimension reduction techniques have the drawback of generated subspaces not having any intuitive meaning. A significant advantage of our visual hierarchical dimension reduction approach is that the generated lower dimensional space is not meaningless to users, in that:

- According to the hierarchical dimension cluster tree visualization, the users know what dimensions are displayed in the reduced dimensional displays.
- If the user finds that a cluster contains some dimensions that have nothing to do with other dimensions in the cluster according to their experience, they can manually remove them from this cluster. Thus each cluster could have a clear domain-specific

practical meaning. This way users can name the representative dimensions with a "meaningful" name, thus helping to interpret the visualization in the reduced dimensional space.

Another significant advantage of our approach is that it integrates automatic and interactive techniques. As a tool to facilitate people in handling data sets, our clustering approach is automatic so that it avoids trivial and boring manual work during the process. At the same time, human interactions are encouraged to make use of the knowledge of the specialists. This combination of automatic and interactive methods is reflected in that:

- Our dimension clustering is an automatic approach, although users can provide their own hierarchical dimension cluster trees instead of using the one generated by our system;
- Users can provide their own similarity calculation routines instead of using the system provided one in the dimension clustering process;
- Users can interactively modify the structure of the hierarchical dimension cluster tree;
- To select clusters from the dimension hierarchical tree, users can employ a combination of automatic and manual brushing mechanisms;
- Users can either select their own representative dimension generating methods, or choose one of the system-provided methods;
- We provide several options to visualize the dissimilarity information of the dimension clusters in the reduced dimensionality displays.

Besides our contribution in visualization of high dimensional data sets, another contribution is that we have developed improvements to the radial space-filling hierarchy visualization technique in the aspects of modification, distortion, coloring and brushing.

Moreover, we have developed a working implementation of this VHDR approach as extensions of XmdvTool system. This implementation goes through the whole VHDR approach, including dimension clustering, interactive Sunburst display and visualization of the mapped data in lower dimensional subspaces using all the existing multidimensional visualization techniques in XmdvTool. This allows us to illustrate that VHDR is practical and compatible with all the existing multidimensional visualization techniques implemented in XmdvTool. This implementation will be incoporated in an upcoming release of XmdvTool 6.0.

Finally, we have conducted several case studies using our implemented system and found some interesting patterns from high dimensional data sets. The case studies prove that our approach is useful for exploring high dimensional data sets.

9.2 Open Questions

There are several open questions for this work, such as:

- What's the maximum number of dimensions and data items our VHDR approach can handle?
- How helpful is this VHDR approach in visualizing high dimensional data sets compared to other dimension reduction techniques such as PCA, MDS and SOM?

We analyze the first problem for the three main steps of our VHDR approach respectively.

First, let's consider the dimension clustering algorithm. Since we use data clusters instead of data items in dissimilarity calculations to scale with large data sets, there is no actual limit on the number of data items we can handle. However, we need to run experiments to check to what extent using data clusters instead of data items will affect

the accuracy of the dimension clustering. Regarding the maximum number of dimensions we can handle, we can compare our dimension clustering algorithm with a data clustering algorithm. Dimensions here play a similar role as data items there. Since millions of data items is trivial for a good data clustering algorithm, we think that there is no practical limit on number of dimensions we can handle, since if a data set contains thousands of dimensions, it is considered to be an extremely high dimensional data set in practice.

Second, let's check the dimension hierarchy visualization. According to our experience, Sunburst has no problem in visualizing a dimension hierarchy composed of a few hundred dimensions. However, when visualizing larger hierarchies, the leaf nodes will be too small to see. However, our interactive tools, such as aggregation, zooming and panning, and distortion can effectively overcome this problem. So we claim that arbitrarily large dimension hierarchies can in general be visualized in the second step.

Finally, we take a look at the mapping step. Since choosing how many dimension clusters to construct the lower dimensional space is completely a decision made by the users, we also argue that there is no real limit on the maximum number of dimensions that can by handled with the VHDR approach. Users are in full control in deciding the lower dimensional subspaces. These subspaces are fully adjustable by them.

Since we can apply this approach to our hierarchical visualization techniques that can scale with large data sets, such as hierarchical parallel coordinates, hierarchical scatterplot matrices, hierarchical star glyphs, and hierarchical dimensional stacking, the number of data items we can visualize is not limited either.

As a conclusion, the first question can be answered by inspection of the strategies we have applied in our solution approach; we find that there is no practical limit on the number of dimensions and data items we can handle using the VHDR approach. However, user evaluations and experiments are of course still needed to validate this claim.

Regarding the second question, our case studies show that VHDR is useful in finding patterns from high dimensional data sets. However, once again, more cognitively designed evaluations are needed.

9.3 Future Tasks

In the future, we plan to implement the following tasks:

- implementing and comparing different dimension clustering approaches;
- applying principal component analysis to generate representative dimensions;
- exploring a dissimilarity representation method that could be applied to most visualization techniques;
- implementing brushing in the reduced lower dimensional subspaces;
- evaluating the VHDR approach by user studies and experiments.

Bibliography

[1] D. Keim, M. Hao, J. Ladisch, M. Hsu, and U. Dayal. Pixel bar charts: A new technique for visualizing large multi-attribute data sets without aggregation. *Proc. of Information Visualization 2001, p. 113-120*, 2001.

[2] D.F. Andrews. Plots of high dimensional data. *Biometrics, Vol. 28, p. 125-36*, 1972.

[3] J.H. Siegel, E.J. Farrell, R.M. Goldwyn, and H.P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery Vol. 72, p. 126-41*, 1972.

[4] H. Chernoff. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association, Vol. 68, p. 361-68*, 1973.

[5] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualization of complex data. *IEEE Computer, Vol. 27(7), p. 57-64*, 1994.

[6] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. *Proc. of Visualization '90, p. 361-78*, 1990.

[7] E.J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association, Vol. 411(85), p. 664*, 1990.

[8] W.S. Cleveland and M.E. McGill. *Dynamic Graphics for Statistics*. Wadsworth, Inc., 1988.

[9] D.A. Keim, H.P. Kriegel, and M. Ankerst. Recursive pattern: a technique for visualizing very large amounts of data. *Proc. of Visualization '95, p. 279-86*, 1995.

[10] J. LeBlanc, M.O. Ward, and N. Wittels. Exploring n-dimensional databases. *Proc. of Visualization '90, p. 230-7*, 1990.

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[12] J. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[13] A. Mead. Review of the development of multidimensional scaling methods. *The Statistician, Vol. 33, p. 27-35*, 1992.

[14] T. Kohonen. *Self Organizing Maps*. Springer Verlag, 1995.

[15] A. Flexer. On the use of self-organizing maps for clustering and visualization. *PKDD'99, p. 80-88*, 1999.

[16] D. Brodbeck, M. Chalmers, A. Lunzer, and P. Cotture. Domesticating bead: Adapting an information visualization system to a financial institution. *InfoVis'97, p. 73-80*, 1997.

[17] S. J. Rose. The sunflower visual metaphor, a new paradigm for dimensional compression. *InfoVis'99, p. 128-131*, 1999.

[18] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. *Proc. of Information Visualization '1995, p. 51-58*, 1995.

[19] B. Hetzler, P. Whitney, L. Martucci, and J. Thomas. Multi-faceted insight through interoperable visual information analysis paradigms. *InfoVis'98, p. 137-144*, 1998.

[20] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. *Proc. IJCNN, p. 413-418*, 1998.

[21] J. York, S. Bohn, K. Pennock, and D. Lantrip. Clustering and dimensionality reduction in spire. *Proc. of the Symposium on Advanced Intelligence Processing and Analysis, p. 73*, 1995.

[22] J. A. Wise. The ecological approach to text visualization. *JASIS, Vol. 50, No. 13, p. 1224-1233*, 1999.

[23] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. *Proc. UIST'90, p. 76-83*, 1990.

[24] J. C. Gower and P. G. N. Digby. *Interpreting Multivariate Data*. John Wiley & Sons Ltd, 1981.

[25] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proc. of IEEE Symposium on Information Visualization, InfoVis'98, p. 52-60*, 1998.

[26] D.A. Keim and H.P. Driegel. Visdb: Database exploration using multidimensional visualization. *Computer Graphics & Applications, Sept. 1994, p. 40-49*, 1994.

[27] M. Ankerst, D.A. Keim, and H.P. Driegel. Circle segments: A technique for visually exploring large multidimensional data sets. *Proc. of Visualization '96*, 1996.

[28] P.C. Wong and R.D. Bergeron. Multiresolution multidimensional wavelet brushing. *Proc. of Visualization '96, p. 141-8*, 1996.

[29] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. *Proc. of Visualization '99, p. 43-50*, Oct. 1999.

[30] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical displays: A general framework for visualization and exploration of large multivariate data sets. *Submitted to Computer & Graphics*, 2001.

[31] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Structure-based brushes: A mechanism for navigating hierarchically organized data and information spaces. *IEEE Visualization and Computer Graphics, Vol. 6, No. 2, p. 150-159*, 2000.

[32] R. Chimera, K. Wolman, and B. Shneiderman. Evaluation of three interfaces for browsing hierarchical tables of contents. *Technical Report Technical Report CAR-TR-539, CS-TR-2620*, Feb. 1991.

[33] D.E. Knuth. Fundamental algorithms. *art of computer pro-gramming, volume 1*, 1973.

[34] G.W. Furnas. Generalized fisheye views. *Proc. of Computer-Human Interaction '86, p. 16-23*, 1986.

[35] A. Bruggemann-Klein and D. Wood. Drawing trees nicely with tex. *Electronic Publishing, 2(2), p. 101-115*, 1989.

[36] S.K. Card, G.G. Robertson, and J.D. Mackinlay. The information visualizer, an information workspace. *Proc. of ACM CHI'91, Conference on Human Factors in Computing Sys-tems, p. 181-188*, 1991.

[37] G. Robertson, J. Mackinlay, and S. Card. Cone trees: Animated 3d visualization of hierarchical information. *Proc. of Computer-Human Interaction '91, p. 189-194*, 1991.

[38] H. Koide and H. Yoshihara. Fractal approaches for visualizing huge hierarchies. *Proc. of the 1993 IEEE Symposium on Visual Languages, p. 55-60*, 1993.

[39] H. Koide and H. Yoshihara. Interacting with huge hierarchies: Beyond cone trees. *Proc. of Information Visualization '95, p. 74-81*, 1995.

[40] F. van Ham, H. van de Wetering, and J.J. van Wijk. Visualization of state transition graphs. *Proc. of Information Visualization 2001, p. 59-66*, 2001.

[41] R. Dachselt and J. Ebert. Collapsible cylindrical trees: A fast hierarchical navigation technique. *Proc. of Information Visualization 2001, p. 79-86*, 2001.

[42] E. Kleiberg, H. van de Wetering, and J. van Wijk. Botanical visualization of huge hierarchies. *Proc. of Information Visualization 2001, p. 87-94*, 2001.

[43] G. Di Battista, P. Eades, R. Tamassia, and I.G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, N.J: Prentice Hall, 1999.

[44] P. Eades. Drawing the trees. *Bulletin of the Institute of Combinatorics and its Applications, p. 10-36*, 1992.

[45] I. Herman, M. S. Marshall, G. Melancon, D.J. Duke, M. Delest, and J.-P. Domenger. Skeletal images as visual cues in graph visualization. *Data Visualization'99, p. 13-22*, 1999.

[46] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of graphs with radial layout. *Proc. of Information Visualization 2001, p. 43-50*, 2001.

[47] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. *Human Factors in Computing Systems, CHI 95 Conference Proceedings*, 1995.

[48] J. Lamping and R. Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing, Vol. 7 No. 1, p. 33-55*, 1996.

[49] T. Munzner and P. Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. *Proceedings of the VRML95 Symposium, ACM SIGGRAPH*, 1995.

[50] T. Munzner. H3: Laying out large directed graphs in 3d hyperbolic space. *Proceedings of the 1997 IEEE Symposium on Information Visualization, p. 2-10*, 1997.

[51] T. Munzner. Drawing large graphs with h3viewer and site manager. *Proceedings of the Symposium on Graph Drawing GD 98, p. 384-393*, 1998.

[52] B.K. Kleiner and J.A. Hartigan. Representing points in many dimensions by trees and castles. *Journal of the American Statistical Associatioin, June 1981, p. 260-272*, 1981.

[53] B. Shneiderman. Tree visualization with tree-maps: A 2d space-filling approach. *ACM Transactions on Graphics, Vol. 11(1), p. 92-99*, Jan. 1992.

[54] B. Johnson and B. Shneiderman. Tree maps: A space-filling approach to the visualization of hierarchical information structures. *Proc. of Visualization '91, p.284-91*, 1991.

[55] J.J. van Wijk and H. van de Wetering. Cushion treemaps: Visualization of hierarchical information. *Proc. of Information Visualization 1999, p. 73-78*, 1999.

[56] K. Andrews and H. Heidegger. Information slices: Visualising and exploring large hierarchies using cascading, semicircular discs. *IEEE Information Visualization Symposium 1998, Late Breaking Hot Topics Paper, p. 9-12*, 1998.

[57] M. Chuah. Dynamic aggregation with circular visual designs. *Proc. of Information Visualization '98, p. 35-43*, 1998.

[58] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualization. *Proc. of Information Visualization 2000, p. 57-65*, 2000.

[59] J. Stasko, R. Catrambone, M. Guzdial, and K. Mcdonald. An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Human-Computer Studies, Vol. 53, p. 663-694*, 2000.

[60] T. Barlow and P. Neville. A comparison of 2-d visualization of hierarchies. *Proc. of Information Visualization 2001, p. 131-138*, 2001.

[61] K. Jain and C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[62] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

[63] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *VLDB'94, p. 144-155*, 1994.

[64] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. *Proc. of KDD '1998, p. 9-15*, 1998.

[65] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *SIGMOD Record, vol.27(2), p. 73-84*, June 1998.

[66] A. Hinneburg and D. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. *VLDB'99*, 1999.

[67] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record, vol.25(2), p. 103-14*, June 1996.

[68] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *Proceedings of ACM SIGMOD98 International Conference on Management of Data, p. 94-105*, 1998.

[69] M.O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. *Proc. of Visualization '94, p. 326-33*, 1994.

[70] A.R. Martin and M.O. Ward. High dimensional brushing for interactive exploration of multivariate data. *Proc. of Visualization '95, p. 271-8*, 1995.

[71] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Navigating hierarchies with structure-based brushes. *Proc. of Information Visualization '99, p. 58-64*, Oct. 1999.

[72] M. O. Ward, J. Yang, and E. A. Rundensteiner. Hierarchical exploration of large multivariate data spaces. *Proc. Dagstuhl Seminar on Scientific Visualization*, 2000.

[73] J.O. Kim and C. W. Mueller. *Introduction to factor analysis: What it is and how to do it*. Newbury Park: Sage Publications, 1978.