DESIGN OF INFORMATION VISUALIZATION AND CASE STUDIES

by

Jie Hao

APPROVED BY SUPERVISORY COMMITTEE:

_____

Kang Zhang, Chair

_____

Balakrishnan Prabhakaran

_____

Xiaohu Guo

_____

Yang Liu

Dedicated to my parents Jinxing Hao and Zhihui Zhou

DESIGN OF INFORMATION VISUALIZATION AND CASE STUDIES

by

JIE HAO, B.S., M.S.

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

August, 2010

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

PREFACE

This dissertation was produced in accordance with guidelines which permit the inclusion as part of the dissertation the text of an original paper or papers submitted for publication. The dissertation must still conform to all other requirements explained in the "Guide for the Preparation of Master's Theses and Doctoral Dissertations at The University of Texas at Dallas." It must include a comprehensive abstract, a full introduction and literature review and a final overall conclusion. Additional material (procedural and design data as well as descriptions of equipment) must be provided in sufficient detail to allow a clear and precise judgment to be made of the importance and originality of the research reported.

It is acceptable for this dissertation to include as chapters authentic copies of papers already published, provided these meet type size, margin and legibility requirements. In such cases, connecting texts which provide logical bridges between different manuscripts are mandatory. Where the student is not the sole author of a manuscript, the student is required to make an explicit statement in the introductory material to that manuscript describing the student's contribution to the work and acknowledging the contribution of the other author(s). The signatures of the Supervising Committee which precede all other material in the dissertation attest to the accuracy of this statement.

ACKNOWLEDGEMENTS

Many people have provided me with inspiration and encouragement that led to the completion of this dissertation.

First and foremost, I owe my deepest gratitude to my Ph.D. program advisor, Dr. Kang Zhang for his support and encouragement in my research from inception to completion. I never could have completed this dissertation without him. Over the years he has provided me with helpful instruction and guidance, excellent research and cooperation opportunities, and great research environment and facilities.

It is an honor for me to have Dr. Xiaohu Guo, Dr. Yang Liu, and Dr. Balakrishnan Prabhakaran as my dissertation committee members. I am grateful for their valuable comments on my research.

I would also like to extend my appreciation to the entire staff of the Department of Computer Science at the University of Texas at Dallas for providing me the administrative support which has allowed me to focus on my research.

The dissertation would not have been possible without the continuous consideration and support of my friends and colleagues. I would also like to thank those who helped with reviews of my research, in particular, Dr. Pushpa Kumar and Chunying Zhao.

Finally, I particularly thank my father, Jinxing Hao, my mother, Zhihui Zhou and my wife, Kun Ding, for their endless love and continuous encouragement.

June, 2010

DESIGN OF INFORMATION VISUALIZATION AND CASE STUDIES

Publication No. _____

Jie Hao, Ph.D.
The University of Texas at Dallas, 2010

Supervising Professor: Kang Zhang

Information visualization (infovis) is the interdisciplinary study of visual representation of abstract information space. Infovis design refers to finding a visual metaphor which properly sheds light on the underlying data. The process of constructing an infovis technique is usually divided into the design stage, in which an infovis technique is designed, and the evaluation stage, in which the designed infovis technique has been implemented and then is enhanced according to the evaluators' feedback. This dissertation provides a guideline called "5 Steps of Designing an Infovis Technique (5DITS)" for the design stage. In this dissertation, 5DITS is applied to design three infovis techniques.

The first infovis technique, called Radial Edgeless Tree (RELT), is designed to visualize hierarchical information on devices with small screens. Few current hierarchy visualization techniques can achieve both desirable readability and efficient screen utilization. RELT solves this problem by recursively dividing a screen into non-overlapping polygons, each of which presents a node. Instead of edges between nodes, nodes' relative positions are used to present

connectivity. RELT has been successfully tested in stock market visualization and simulation of a cell phone browser.

InfoShape, the second infovis technique, provides a global view for multi-dimensional information, which is organized in the table format. InfoShape consists of Record InfoShape (RInfoShape) and Dimension InfoShape (DInfoShape) that visualize multi-dimensional information as a 3D sphere whose appearance denotes how much the multi-dimensional information satisfies a pre-defined set of criteria. By comparing the shapes of different multiple information sets, global content similarities and differences can be quickly captured. RInfoShape and DInfoShape are evaluated on a Java program evaluation and US life table comparison, respectively.

SciTrend, the last infovis technique, visualizes the popularity trend of several computer science research areas and relationships among them. The underlying data is a journal citation network which contains both hierarchical relation and additional non-hierarchical connectivity. The relationships between scientific area and research articles are hierarchical relationships. The citation relationships between research articles are non-hierarchical relationships. SciTrend provides static visual pattern and associated interactive functions, so that users can quickly capture the research areas' popularity trends and relationships, and explore citation details among peer-reviewed journals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Information Visualization

Visualization research was initially started to meet the strong requirements of handling large amounts of scientific data generated by the scientific community [44]. This kind of visualization is usually termed "scientific visualization." For the last two decades, the non-scientific data has been generated by various data collecting applications, instruments, and especially the rapidly-growing internet. Information visualization (infovis) evolved from the scientific visualization to meet the need for non-scientific information exploration and comprehension.

Information visualization, which was first termed by Robertson, Card and Mackinlay [105] in 1989, is formally defined as "the use of computer-supported, interactive visual representations of abstract, non-physically based data to amplify cognition" [23]. By arguing "Information visualization enables mental operations with rapid access to large amounts of data outside the mind, enables substituting of perceptual relation detection for some cognitive differencing, reduces demands on user working memory, and enables the machine to become a co-participant in a joint task, changing the visualization dynamically as the works proceeds" [125], Ware regards information visualization as human external cognition, that is, something outside the mind that can boost the human cognitive capabilities.

Both scientific visualization and information visualization generate visual representations which assist users to gain insights into the underlying data. Although scientific visualization and

information visualization are defined separately, they are not exclusive but are related to each other.

The data in scientific visualization represent objects or concepts associated with some phenomena in the real world. Therefore, these scientific data express structures in 1D, 2D or 3D geometry and essentially have spatial or temporal dimensions. Scientific visual representations usually map the scientific data into the specific given structures. On the other hand, non-scientific data in information visualization have no clear geometric structures. These non-scientific data usually represent abstract relationships and concepts which may not have counterparts in the real world.

Non-scientific data are translated into a data type (also termed as data format or data model) before they are mapped to a graphical representation. The data type is usually considered as a starting point for designing information visualization techniques. A classic way of summarizing data types was proposed by Shneiderman [114]. Seven types of data are summarized: linear data, 2D data, 3D data, hierarchical data, network data, multi-dimensional data and content-based data.

1.2 Two Stages of Constructing an Infovis Technique

Plaisant asked a very provocative question in his paper:

"Is a Segway a better vehicle?" [97]

In the real world, this question cannot be answered because of the uncertainty of the following questions:

"Better than what? For whom? To go where? In which state of mind? How important is it to get there quickly, or to get there at all? What if you do not really know where you are going? What is the best vehicle then? Should I use my savings to buy a Segway when

I have other needs? Usability studies and formal comparisons of speed characteristics and incident data might help worried potential drivers but it is their judgment of utility that will likely trigger adoption." [97]

This situation also occurs when we ask the question: how to construct a good infovis technique? This general question leads to additional questions such as: What is the real problem that needs to be solved? Is the problem actually solvable? Who are the users? Do any specifications exist considering users' knowledge, background, and awareness of the problem and so on? What does the underlying data look like? Can any existing infovis techniques be utilized? Are there any performance records about these existing infovis techniques?

The process of building an infovis technique can be divided into two major stages: the design stage and the evaluation stage. During the design stage, an infovis technique is designed and detailedly documented, but has not been implemented. In the evaluation stage, an infovis technique is mostly implemented. User-based empirical studies are usually conducted in the evaluation stage and an infovis technique is improved based on users' feedback.

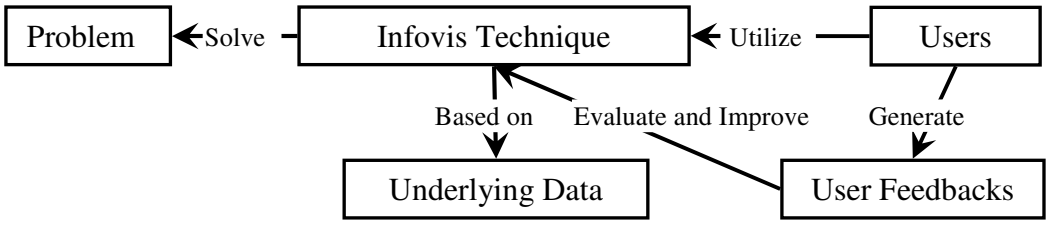


Figure 1.1. Essential Idea of Research in the Evaluation Stage

Much researches has been done for the evaluation stage. Figure 1.1 illustrates the essential idea of research in the evaluation stage: users utilize an infovis technique, which is based on particular type of underlying data, to solve a list of tasks. With this technique, users generate some feedback that is further applied to evaluate and improve the infovis technique. Current research topics on the evaluation stage involve establishing metrics and benchmarks of visual

quality, selecting representative participants, choosing appropriate tested tasks, minimizing bias in the evaluation and so on. These studies are driven by usability factors such as data, user and task and contribute much to understanding an infovis technique's potential and limitations, comparing various implemented visualization techniques, and identifying possible improvements. In short, these studies show the power of improving an infovis technique. Current research in the evaluation stage will be introduced in next section.

Designing an appropriate infovis technique is as important, if not more important, than evaluating and improving an implemented infovis technique. An appropriate way of designing an infovis technique can lead to a technique that has a high potential of further improvement in the future evaluation stage; on the other hand, an inappropriate way very possibly leads to a technique which can hardly be enhanced after being implemented. The following metaphor demonstrates the importance of and relationship between the design stage and the evaluation stages: Imagine that we are going to select a seed pigeon in the design stage and train it to be a homing pigeon in the evaluation stage. Before starting to train a pigeon, then, we have to make sure it is not a chicken!

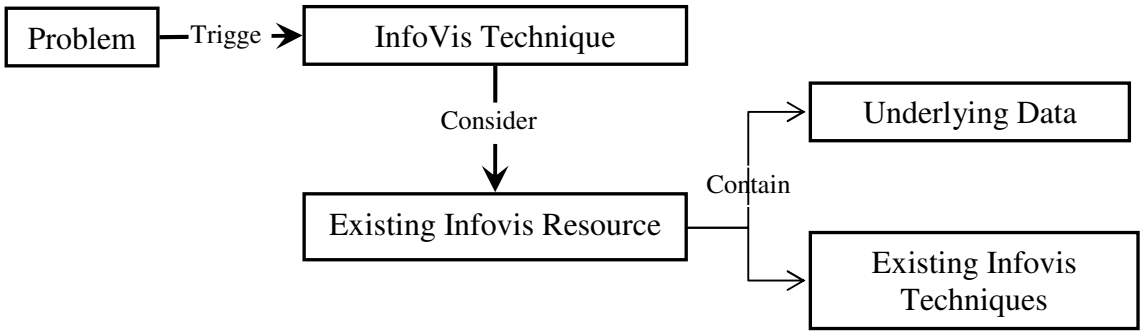


Figure 1.2. The Idea of Designing an Infovis Technique in the Design Stage

This dissertation contributes on a methodology for "selecting a good pigeon,", or, in other words, a methodology for designing an appropriate infovis technique in the design stage. Figure 1.2 illustrates the basic idea in the design stage: a problem triggers the demand for an infovis

technique and existing visualization resource needs to be considered. Unlike the evaluation stage, the process of designing a visualization technique in the design stage is problem-driven.

1.3 Related Work

Infovis technique design does not end at invention or implementation. As more experience can be gained with the evaluation of infovis techniques, much research has been done on how to complete a proper evaluation.

Evaluation methods are generally classified into inspection methods and testing methods [4].

Inspection methods refer to heuristic methods in which evaluators inspect an infovis technique using their experience and analysis, rather than empirical study. Many inspection methods can be applied to infovis techniques [90] that have not yet been implemented.

Nielsen and Mack divide inspection methods into five groups [90]:

- Heuristic Evaluation: A small group of people evaluate an infovis technique using a given checklist of general principles and use their judgment to generate a list of potential problems [88] [91] [89].
- Guideline Checking: An individual inspects an infovis technique by checking it against a list of very detailed and descriptive principles and reports how much the inspected infovis technique deviates from each principle. Example guidelines can be found in [115] [17] [83].
- Cognitive Walkthrough: This type of inspection method, first introduced in [80], focuses explicitly on learnability. A small group of individuals simulate a novice user's mind and walk through certain tasks. At every step along the correct path, simulated success or failure stories are recorded [81] [64] [127] [118].

- Guideline Scoring: An evaluator inspects an infovis technique against a list of specific principles. Each principle is scored and a total score is calculated to present the degree to which the infovis technique satisfies the list of principles [92].
- Action Analysis: This type of method aims at evaluating efficiency. An evaluator breaks a task down into several smaller steps and estimates the time that an expert would take to complete each step. The total amount of time for completing the entire task is computed by summing up the atomic action time [81] [102] [22].

Testing methods refer to techniques in which individuals use an infovis technique to complete tasks so that evaluation can be made based on user feedback. There are three main types of testing methods: formative tests, summative tests, and usage tests [5].

- Formative tests: A small group of evaluators test an infovis technique by completing a set of tasks so that they can capture the problems and understand why these problems occur [76]. The classic type of formative test is the thinking aloud test, in which users are asked to explicitly verbalize what they are doing, what information they see, what questions they have, any confusion they might experience, and what decisions they make when they are using an infovis technique. A formative test shows its value in the software development circle.
- Summative tests: A large group of evaluation users complete a set of tasks using an infovis technique so that information of pre-defined measurement parameters can be collected. Measurement parameters contain task completion time, number of clicks and so on. For a more detailed list of measurement parameters, refer to [4]. A statistical analysis is usually conducted on the cumulated measurement information [40] [76]. The classic type of summative test is the formal experiment. Formal experiments are usually used for two major purposes: 1) to test the absolute performance of an infovis technique, and 2) to compare

alternative infovis technique designs. There are two ways of designing a formal experiment to compare different infovis techniques: independent measures (also called between-group) and repeated measures (also called within-group). Independent measures involve two equally-sized groups of users, and each group tests only one infovis technique by completing identical tasks. Repeated measures, however, only involve one group of users who are divided into two equally-sized pools. Users in two pools test all infovis techniques in a counterbalanced order. For example, two infovis techniques, A and B, are to be evaluated. In independent measures, the group X only tests A and the group Y only tests B. In repeated measures, users in pool M first test A and then test B, and users in pool N first test B and then test A. Formal experiments usually require a large group of users so that a statistical difference can be demonstrated. Andrews [4] claims that he rarely finds statistical difference in collected performance data if the number of users ranges from 1 to 32.

- Usage tests (observational studies): This type of evaluation involves observation of individuals performing various tasks while using an infovis technique over a long period of time, usually from several hours to several days [57] [72] [73]. Usage studies mostly depend either on self-reporting where users manually produce a log recording their activities while using an infovis technique [24] [112] or on automatically-generated logs, which are later manually transcribed into users events [20] [21]. Each method has its tradeoffs. Self-reporting is simple to design, but is often criticized due to its unwarranted accuracy. Although automatically-generated logs minimize the risk of inaccuracy, manual coding from logs to user events is very time-consuming [20] [21]. Usage studies are usually applied to learn how individuals use an infovis technique; however, they cannot be used to compare different infovis techniques. For more usage studies on information visualization, refer to [55].

All aforementioned research works show their power on various aspects of the design for infovis techniques. However, most of them are in the evaluation stage, and none of them can yield a guideline for designing an infovis technique. This dissertation provides a guideline that designers can follow when they are in the design stage of infovis design.

1.4 From Business Consulting to Designing an Infovis Technique

1.4.1 Similarities

Designing an infovis technique is not only about creativity, but also about analyzing priority, balancing tradeoffs and so on. For example, an infovis technique, possibly consisting of several functionalities, aims at solving a problem as a one-piece unit; overly improving a single minor functionality will most likely decrease the performance of the entire infovis technique. Therefore, working through the design stage requires strategic thinking.

Following are some questions that may be asked in the design stage:

1. What is the problem that needs to be solved?
2. How can a systematic view of the problem be developed?
3. Does any major parameter of this problem require more attention?
4. How can the existing resources be maximized?
5. Based on existing resources, how are improvements identified?
6. How will conflicts be handled?
7. How will a quantitative assessment of the designed infovis technique be obtained?

Designing an infovis technique and strategic business consulting share some major issues:

- Many visualization techniques and consulting tools have been invented, which systematically cover most, if not all, of the topics in both infovis areas and business consulting fields. For

convenience and cost reduction, both situations require squeezing the maximum utility out of every available resource.

- Both an infovis technique and a business case are triggered by a particular problem. For infovis, a triggering problem would be: reduce the number of edge crossings, enhance the node-link diagram readability, and visualize Starbucks sells data in Dallas-Fort Worth area. In business strategic consulting, common triggering problems involve: double the sells next quarter, enhance the efficient of current human resource structure, and develop a system of internal branding and so on.
- Both have the same purpose: all efforts aim at providing a well-suited solution that best solves the triggering problem.

Based on the above similarities, the key issues faced in the design stage overlap with the basic issues faced in strategic consulting. This dissertation introduces a guideline for designing an appropriate infovis technique for a triggering problem. The guideline is adapted from a widely used strategic consulting tool due to the aforementioned similarities.

1.4.2 A Strategic Problem Solving Tool: Mckinsey's 7 Steps

Life consists of problems stacked upon one another. We face different types of problems every day. Some are minor, such as losing sunglasses before leaving on a cruise trip or when the assembly line in a factory does not work simply because a screw is loose. Major problems also occur, although they tend not to happen as frequently as minor problems. Major problems may include deciding how to develop career over the next ten years or how to optimize the human resource structure of a large company, such as General Electric. All problems, from minor to major, require strategic thinking.

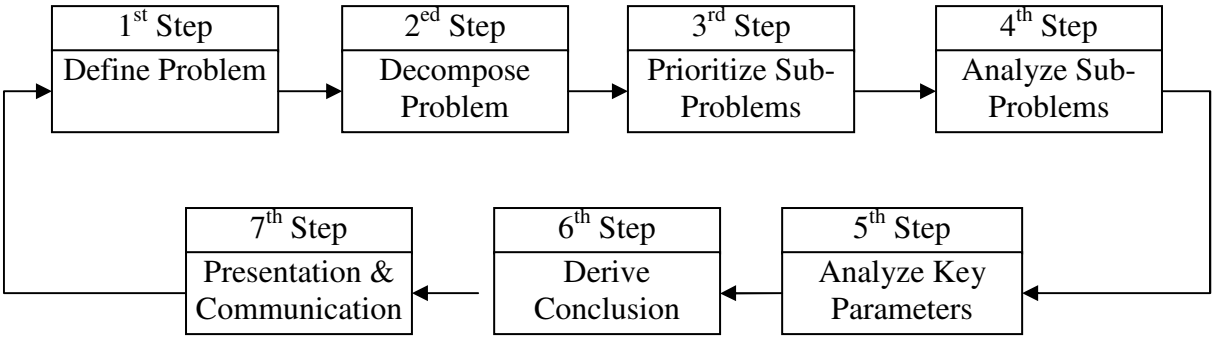


Figure 1.3. Mckinsey's 7 Steps

Mckinsey's 7 Steps illustrated in Figure 1.3 form a popular strategic consulting tool that provides a problem-solving methodology, which (as per the name) consists of seven steps:

1. Define the problem. This key step aims at specifying a clear scope of the problem and eliminating ambiguity.
2. Organize the problem defined in the design stage in a more structural manner. Mckinsey provides several methods for decomposing a problem into a hierarchy called a logic tree. A logic tree provides an opportunity to think about and view a problem systematically. Moreover, the last level of a logic tree is composed of concrete, manageable sub-problems. The sub-problems can be conquered one by one without losing the concept of the whole problem.
3. Sort the sub-problems at the last level of a logic tree, usually according to importance and urgency. Key sub-problems rank high and are paid more attention; sub-problems with low ranking are attached less importance or even dropped.
4. Analyze sub-problems (associated with detail scheduling and setting up hypotheses). In this step, sub-problems are usually assigned (with a time limitation) to team members.

5. Analyze the key parameters of hypotheses and identify their validity. If one hypothesis is proven invalid, another hypothesis should be set up.
6. Draw a conclusion for each sub-problem, and derive an overall conclusion for the whole problem according to the logic tree.
7. Properly present the conclusion and communicate with clients.

Mckinsey's 7 Steps provide a standard mental process. This kind of problem-solving tool is required when a system or a technique needs to move from a given state to a desirable goal state [151]. This dissertation adapts Mckinsey's 7 Steps to a guideline of designing an infovis technique.

1.5 Outline of the Dissertation

This dissertation is organized as follows: This chapter introduces background knowledge of infovis, two stages of constructing an infovis technique and current related research. A problem-solving tool, Mckinsey's 7 Steps, is introduced followed by a discussion of similarities between designing an infovis technique and consulting a business case. CHAPTER 2 adapts Mckinsey's 7 steps to the guidance of designing infovis technique and generates a guideline called 5 Steps of Desinging an Infovis Technique (5DITS). CHAPTER 3, CHAPTER 4 and CHAPTER 5 discuss the infovis techniques that are designed following 5DITS. CHAPTER 3 introduces a new infovis technique called RELT which visualizes and navigates a hierarchy on small screens. CHAPTER 4 discusses InfoShape, which visualizes multi-dimensional data as 3D sphere so that global similarities and difference among information sets can be quickly understood. CHAPTER 5 introduces SciTrend, which users can utilize to understand the popularity trend of several computer science research areas and relationships. CHAPTER 6 summarizes the research in this dissertation.

CHAPTER 2

FIVE STEPS OF DESIGNING AN INFOVIS TECHNIQUE

CHAPTER 1 introduced Mckinsey's 7 steps. Based on infovis design needs, the number of steps has been reduced from seven to five. Figure 2.1 illustrates the 5DITS. Each step is discussed in detail and is illustrated by an example.

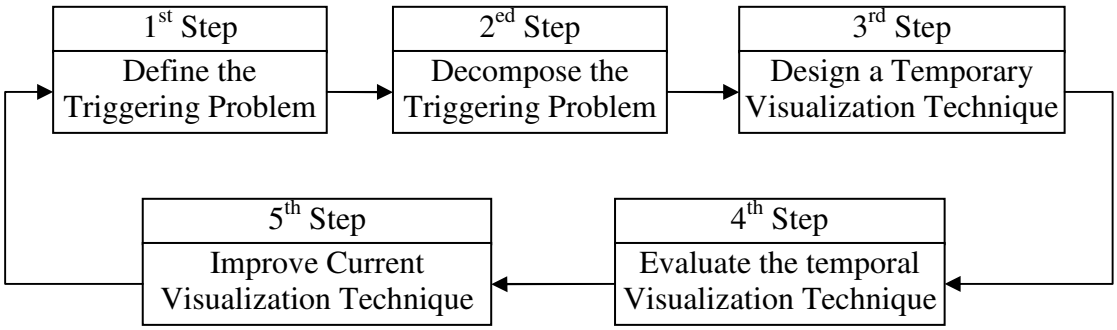


Figure 2.1. Illustration of 5DITS

2.1 Step One: Define the Triggering Problem

Einstein is widely quoted as having said that if given only one hour to save the world, he would spend fifty-five minutes defining the problem and only five minutes finding the solution. This quote emphasizes the problem-solving mantra: find the right thing, and then do things right. Before jumping into finding a solution, it's worth investing the necessary time and effort to gain deep insight into the real problem.

The need for an infovis technique is usually triggered by a problem that needs visual representation to enhance comprehension. Defining the problem is the key step because it helps us understand what the real problem is. It serves as a compass to avoid deviation from the

original purpose. Also, it aids in achieving consensus among those who may describe the problem in terms of their different perspectives. A triggering problem has no limitation. It could be either as concrete a problem as reducing the number of edge crossing in a node-link diagram, or as abstract as creating a better infovis technique for visualizing a hierarchy. This dissertation uses the symbol *TP* to denote the term triggering problem. Triggering problem definition should follow three principles:

- S1P1: Find the real *TP*. The real *TP* very likely exists under the guise of different symptoms that reveal certain aspects of the *TP*. For example, a *TP* may be incorrectly defined as increasing the clarity of a node-link diagram when this may simply define a symptom of the real *TP* which is reducing the number of edge crossings.
- S1P2: Keep clear and concise. The definition of *TP* needs to be short and concise. Although there is no strict requirement for the length of a definition, usually a one-sentence description is desirable.
- S1P3: Use appropriate scope. A narrow definition of *TP* may lead to an infovis technique which only partially solves the real *TP*. A definition with a too wide a scope may lead to functional redundancies within an infovis technique.

2.1.1 Example

Abundant data, such as text, DNA and code, exist in the form of a string which has an inherent structure. An infovis technique showing important structural features is needed to accelerate users' understanding of string data. One important structural feature of string data is the repetition of sub-strings. Melodies, for example, are easily distinguished by repeated musical passages. Therefore, a natural way to visualize the structure in string data is to highlight these repeated sub-string patterns.

Wattenberg [126] proposed an infovis technique, arc diagrams, to represent the complex patterns of repetition in string data. This section applies the 5DITS to simulate the process of designing arc diagrams.

The title of Martin's paper explicitly defines *TP* as visualizing structure in strings.

2.2 Step Two: Decompose the Triggering Problem

A *TP* may be fairly complex, and the expected corresponding infovis technique is very likely equally complex. A traditional method of handling this complexity is to decompose *TP* into a collection of sub-problems whose complexity is relatively small and can be solved more simply. As illustrated in Figure 2.2, the sub-problems compose a logic tree, which provides a method for solving problems systematically. Via the reverse process of composition, the solutions to sub-problems can ultimately be combined to arrive at a solution for the entire *TP*.

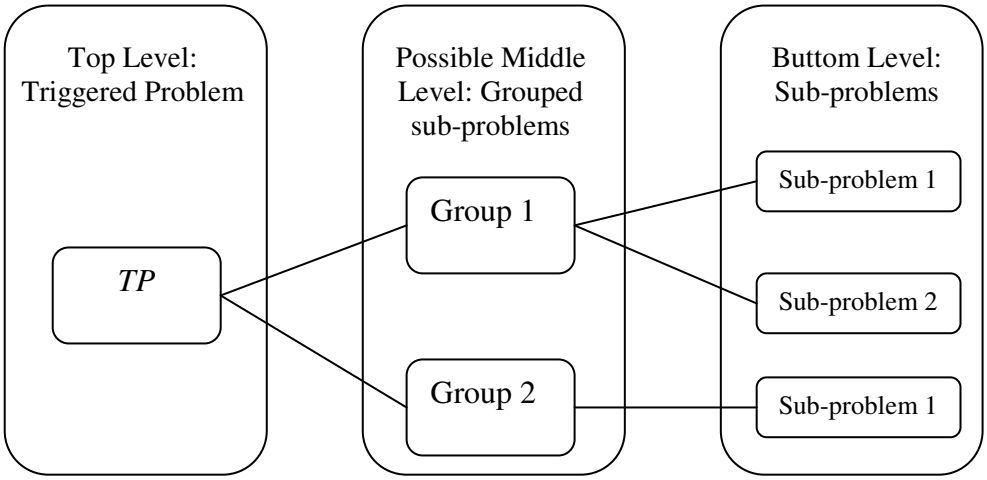


Figure 2.2. An Example Structure of a Logic Tree

After establishing a logic tree, the sub-problems are assigned weight according to their importance to the *TP*. This system focuses our attention on key sub-problems. No fixed criteria exist to construct a logic tree for a *TP*. Even for one *TP*, there may be different ways of

organizing the sub-problems. Four principles should be followed during the process of breaking down a *TP*:

- S2P1: Follow the Mutually Exclusive, Cumulatively Exhaustive (MECE) principle. MECE, introduced by Rasiel [101], is the hallmark of problem solving per Mckinsey. MECE is a widely-used grouping principle, in which a problem is divided into sub-problems that comprehensively describe the problem without overlapping. Strictly following MECE, then, ensures a designed infovis technique that completely solves the *TP* without any redundant functionality.
- S2P2: Keep the logic tree structure compact. The possible middle level illustrated in Figure 2.2 is used to group sub-problems which serves related functionalities. This level is necessary for an abstract *TP,* such as creating a better infovis technique for hierarchical information. For a concrete *TP,* such as reducing the number of crossings for a node-link diagram, the middle level can be omitted. Usually, keeping the number of levels under four is optimal. In addition, a reasonable number of nodes in a group (if one exists) should be between two and five.
- S2P3: Use concrete and manageable sub-problems. Make sure that every sub-problem at the bottom level is concrete and solvable.
- S2P4 Avoid going into too much detail. Mathematician George Polya's book, *How to Solve it*, clearly justifies this principle:

  "If you go into too much detail you may lose yourself in details. Too many or too minute particulars are a burden on the mind. They may prevent you from giving sufficient attention to the main point, or even from seeing the main point at all. Think of the man who cannot see the wood for the trees. Of course we do not wish to waste our time with

unnecessary detail and we should reserve our effort for the essential. The difficulty is that we cannot say beforehand which details will turn out ultimately as necessary and which will not." [99]

2.2.1 Example

Figure 2.3 illustrates a logic tree of *TP*: visualizing structure in strings. Wattenberg [126] mentions all sub-problems in this logic tree in his paper and suggests that an appropriate infovis technique should solve them all.

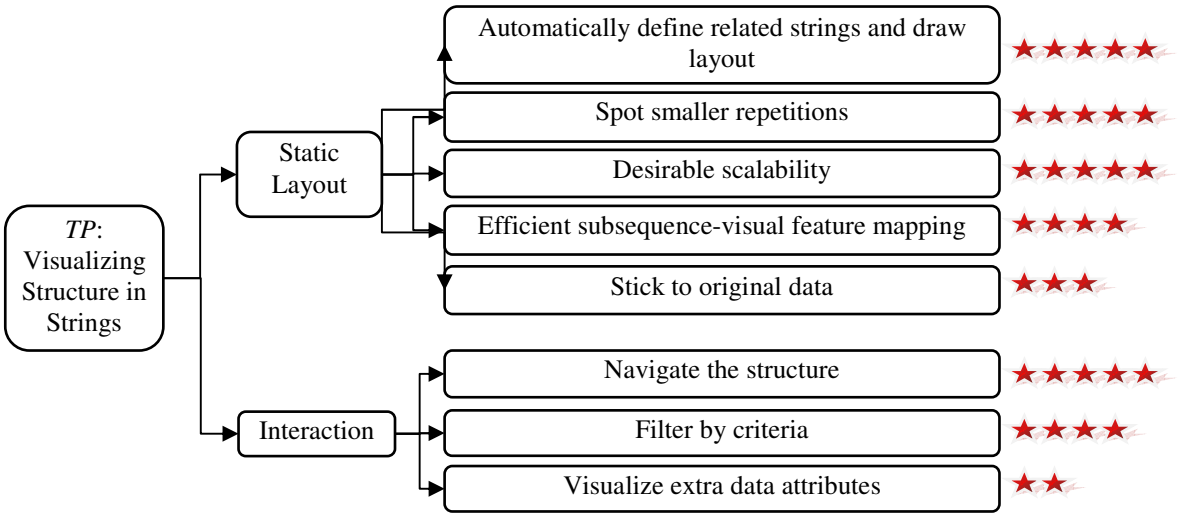


Figure 2.3. An Example Logic Tree

These broken-down sub-problems are divided into two groups: static layout and interaction. Sub-problems in static layout group refer to considerations of necessary features of the static layout generated by the expected infovis technique. Sub-problems in interaction group mainly consider the functionalities that should be supported to dynamically explore the structure in a string. Sub-problems in the same group are ranked according to their weight which is denoted by the number of stars.

Static Layout Group:

- Automatically define related sub-strings and draw the layout.

  Automation in this sense includes two components: automatically identifying related sub-strings and automatically generating a layout. Automation is the most important advantage of using a computer and is usually a very basic requirement of an infovis technique. Wattenberg [126] emphasizes automation when he discusses the disadvantages of Schenkerian diagrams [111] and TimeSketch [155] which provide natural ways of showing structure, but are unsuitable for automation.

- Spot smaller repetitions.

  String data may contain several significant repeated subsequences composed of even smaller repetitions. Melodies, for example, are usually based on a combination of large musical passages consisting of smaller rhythms. Wattenberg suggests that smaller repetitions are as important as larger ones. [126]

- Desirable scalability.

  Limited scalability could occur for several reasons. Schenkerian diagrams are not easily scalable because they can neither draw a layout automatically nor show structural features at different levels. TimeSketch does not scale well for sequences having many related passages because it requires human definition of related passages and uses color coding to indicate related passages. An appropriate infovis technique should be easily scalable for strings that have complex structures and contain many instances of the same sequence.

- Efficient subsequence-visual feature mapping.

  Dotplot [29] is discussed as a counterexample in terms of efficient subsequence-visual feature mapping. Although Dotplot can handle huge data sets, its matrix graphical representation is fairly inefficient, i.e., if there is a subsequence repeating *n* times; the

corresponding visual feature will rise to n^2 times. This quadric scaling problem tends to produce a confusing layout when Dotplot is applied to a string with frequently repeated sub-strings. An appropriate infovis technique should have efficient subsequence-visual feature mapping.

- Stick to original data.

  It is unacceptable to visualize the string structure at the cost of eliminating certain portions of original information. Chaos Game [62], for instance, visualizes the frequencies of various motifs by sacrificing the ordering information of a string. This means that Chaos Game cannot be applied to a domain where order is important.

Interaction Group:

- Navigate the structure.

  Users should be allowed to explore a string's structure from a very high level (denoting the entire string) to a lower level (depicting repeating subsequences).

- Filter by criteria.

  An appropriate infovis technique should allow a user-assigned criterion. For example, when users specify length, only the subsequences having the specified length are highlighted.

- Visualize extra data attributes.

  Although the major purpose is to visualize the structure in a string, including other important attributes into the visualization could be an added value. For example, for a string of numbers, being able to embed the numbers into the visualization makes the infovis technique more helpful.

2.3 Step Three: Design a Temporary Infovis Technique

Because the sub-problems at the last level of a logic tree are manageable, possible methods are assigned to each sub-problem. An existing method is denoted as *EM*. An *EM* could be as simple as encoding a property into a visual feature or as complex as the Treemap or Fisheye techniques.

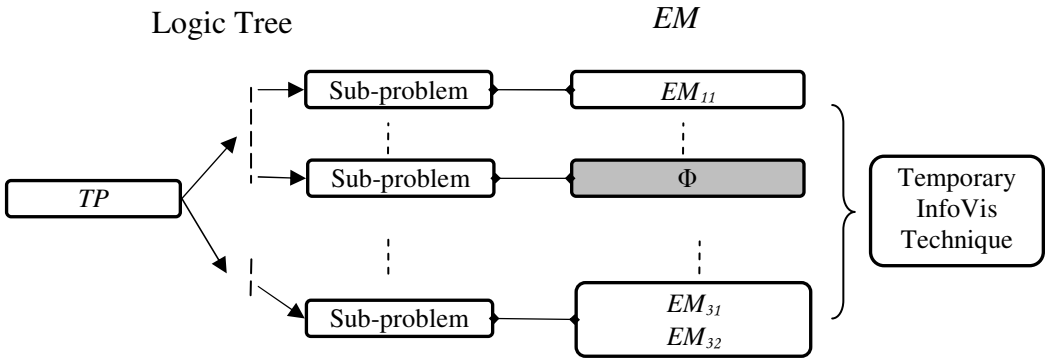


Figure 2.4. Designing a Temporary Infovis Technique

Two principles should be followed when attempting to find sub-problems associated with *EMs*:

- S3P1: If there is no *EM* for a sub-problem so far, then leave the sub-problem unsolved. The gray rectangle in Figure 2.4 illustrates this scenario.
- S3P2: If there are several *EMs* for a sub-problem, then simply list them. The lower right rectangle in Figure 2.4 shows two *EMs* for the third sub-problem. The first number in the subscript of an *EM* denotes the identity of the associated sub-problem, and the second number in the subscript indicates the order of the *EM*.

After assigning each sub-problem possible *EMs*, select one *EM* for each sub-problem (if one exists) and compose these *EMs* into a temporary infovis technique. A temporary infovis technique, then, can be represented as follows:

$$\text{Temporary Infovis Technique} = \{EM_{ij} \ldots EM_{xy}\}$$

Equation 2.1. Temporary Infovis Technique Expression

2.3.1 Example

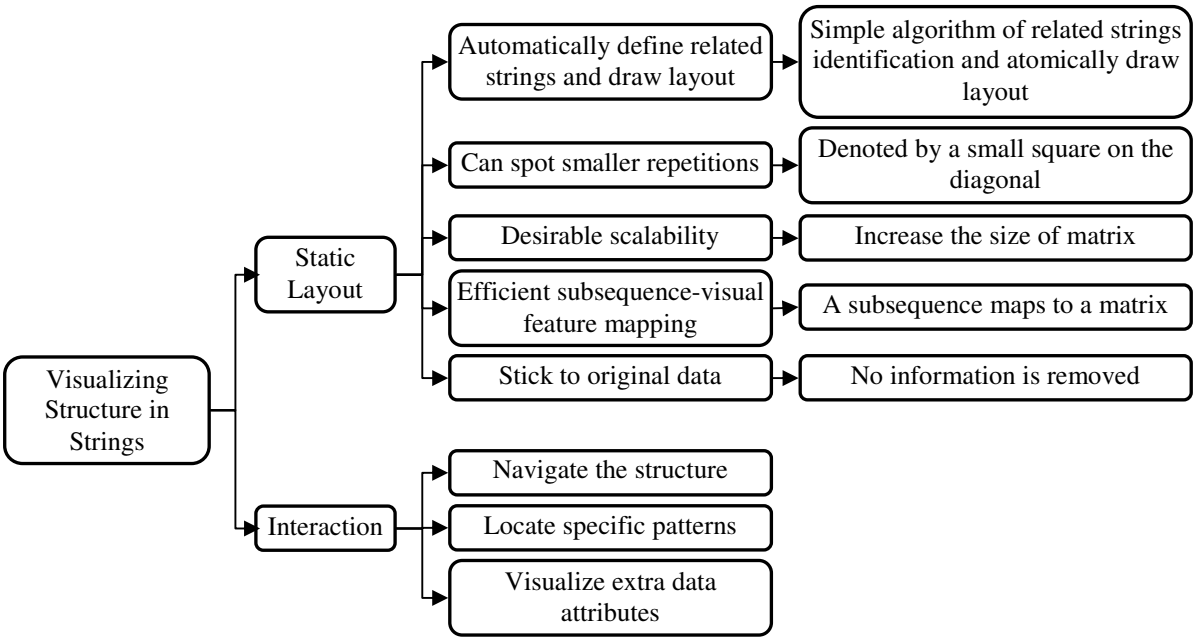


Figure 2.5. A Temporary Infovis Technique

Wattenberg [126] discusses several infovis techniques that aim at visualizing structure in strings. Dotplot [51] is one of the most widely-used solutions. The following figure utilizes Dotplot as a temporary infovis technique to explain how Dotplot solves the *TP:* visualizing structure in strings.

As Figure 2.6 illustrates, Dotplot is based on a simple algorithm: a string of n symbols $a_1\ a_2 \ldots a_n$ is denoted by an n x n matrix-like image, in which the pixel at coordinates (i, j) in the image is black if $a_i = a_j$ and white if not. This algorithm produces a visual auto-correlation matrix illustrating a string's structure, and all sub-problems in the static layout group are solved, although perhaps not perfectly. The sub-problems in the filter interaction group (gray rectangles) are left unsolved. Figure 2.5 illustrates the logic tree of this solution for *TP*: visualizing structure in strings.

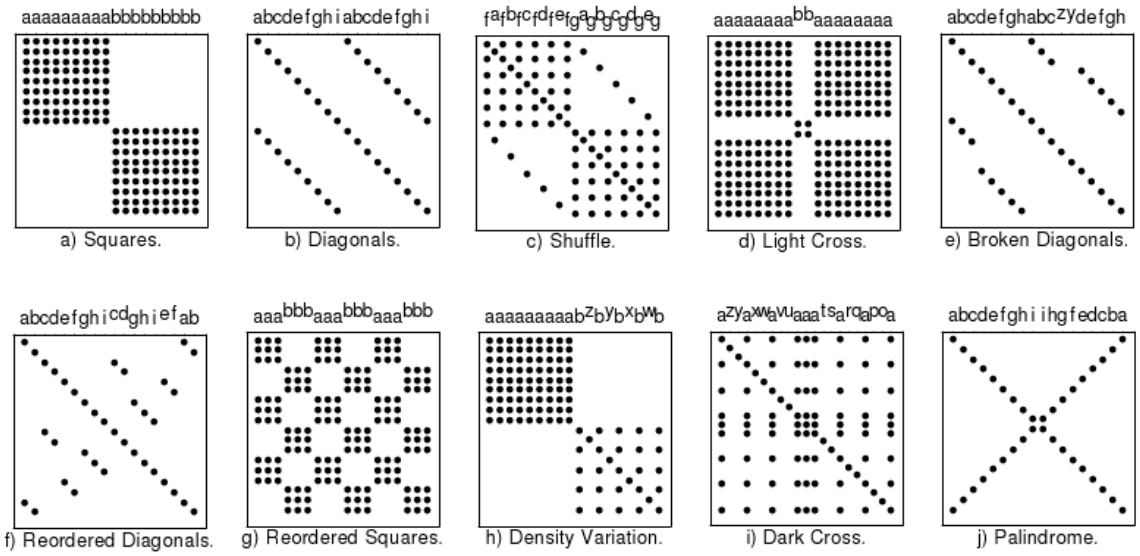


Figure 2.6. Synthetic Dotplots [51]

2.4 Step Four: Evaluate the Temporary Infovis Technique

The temporary infovis technique constructed in the previous steps is a simple combination of *EMs* and hence is very likely to have potential for improvement. Several possible improvements are introduced in the first part of this section, followed by metrics for assessing the performance of the infovis technique.

2.4.1 Possible Improvements

Following are several aspects that usually require evaluation for the possibility of further improvement.

- A sub-problem has no *EM*.

  In this scenario, the temporary infovis technique lacks of the necessary functions to solve certain sub-problems. However, this is a great opportunity to build a new infovis method from scratch.

- A sub-problem cannot be perfectly solved by any *EM*.

This scenario frequently occurs in visualization design because most *EMs* are not constructed for purely theoretical problems. Rather, they are constructed for specific problems, which invovles a variety of complicated real world situations such as data, display equipment, or users' esthetic preference. This fact creates many possibilities for improving an *EM* and making it more suitable for its own sub-problem.

- *EMs* conflict.

  Pay attention to conflict *EMs*. A pair of two conflict *EMs* cannot desirably solve their associated sub-problems at the same time: one EM will be used at a detriment to the other. For example, if a certain layout is used to show the information distribution, and meanwhile, the fisheye technique is applied to accelerate the comprehension of the local details, then the layout will inevitably be distorted when the fisheye is applied. Therefore, the *EM* generating the layout and the fisheye form a pair of conflict *EMs*. As the old adage says, you cannot have your cake and eat it too. There may exist hundreds of pairs of conflicting EMs, and this guide cannot explain how to deal with conflicts one by one. In general, assigning weight to sub-problems (see Step Two) can be used to handle conflicts. A sub-problem with more weight should be solved with priority. For example, if illustrating information distribution has more weight than viewing local details, then the *EM* generating layout gains much more significance than the fisheye technique. In this case, a visualization method that illustrates local details without distorting the overall layout may be chosen instead.

2.4.2 Evaluation of an Infovis Technique

Unlike some implemented infovis techniques in the evaluation stage, the infovis technique in the design stage remains unimplemented and, of course, cannot be used for an empirical study. Guideline scoring [92] is applied to evaluate the effectiveness of an infovis technique in the

design stage. As per discussions in previous sections, the process of designing an infovis technique is problem-centered; therefore, how well a *TP* is solved by an infovis technique will directly determine the effectiveness of the infovis technique.

Several concepts and notations need to be introduced in advance.

Degree of Satisfaction, denoted by *DoS*, is a number between 0 and 1 that measures the performance of an infovis technique. A larger *DoS* indicates better performance. *DoS* is computed by Equation 2.2,

$$DoS = \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k}$$

Equation 2.2. *Dos* Calculation

Where w_k denotes the weight of the k^{th} sub-problem; dos_k denotes the degree of satisfaction of *EM* for the k^{th} sub-problem; w_i and w_j indicate the weight of a pair of sub-problems solving a pair of conflict *EMs*; and doc_{ij} denotes the degree of conflict between a pair of conflicting *EMs*.

Equation 2.2 proves that designing an infovis technique is a process of balancing the tradeoffs. An appropriate infovis technique may not consist of the best *EMs*. Only two factors can increase *DoS* as a whole: 1) increasing the *dos* for each *EM* and 2) considering the negative impact of conflicts *EMs*.

2.4.3 Example

Figure 2.7 shows an evaluation using Dotplot to solve *TP*: visualizing structure in strings. The stars and percentage denote the weight and *dos* of the associated sub-problems, respectively.

Two sub-problems in the static layout group are only partially solved.

The first one is the sub-problem: spot smaller repetition patterns. Dotplot visualizes the consecutive repeated symbols as a square along the image's diagonal. The inconsecutive

repeated symbols, however, are visualized as two symmetric patterns with respect to the image's diagonal. While Dotplot yields an esthetically pleasing layout, small patterns are very likely to be overwhelmed when the data is very large.

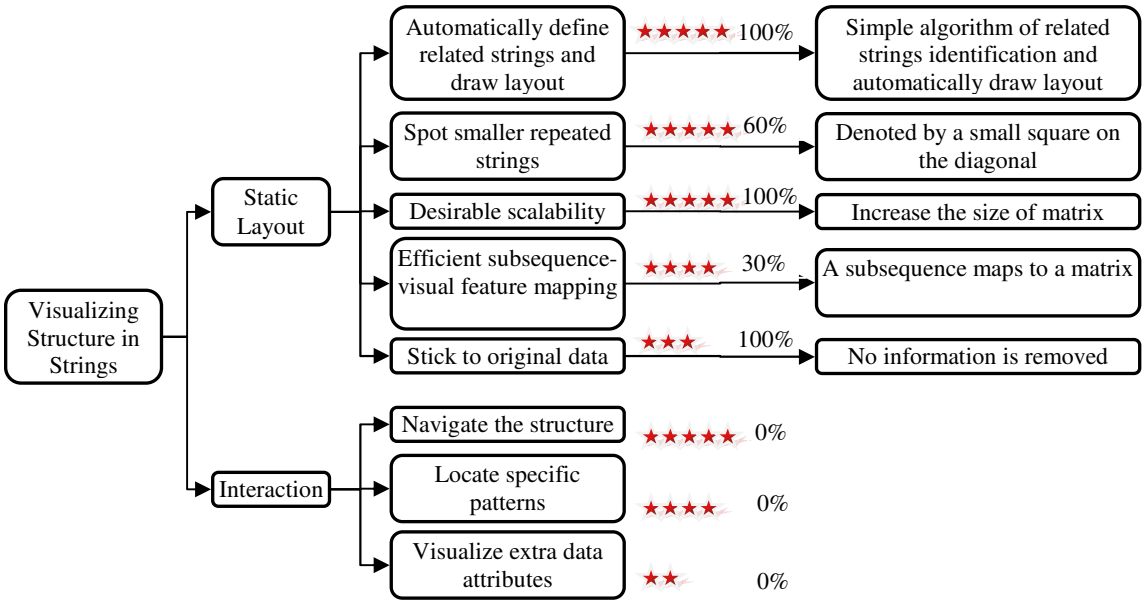


Figure 2.7. Evaluation of Dotplot

Figure 2.8 illustrates a Dotplot layout where approximately 2 million lines of C code are visualized and smaller repetitions are hardly recognizable. In addition, using 2D coordinates to depict repetition contained in 1D string data imposes extra recognition burden on users. Accordingly, the *dos* is only 60%.

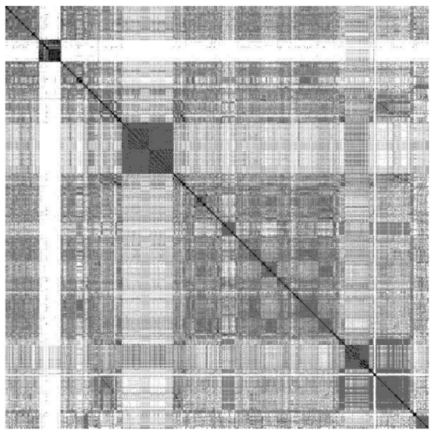


Figure 2.8. Dotplot Example for Approximately 2 Million Lines of C Code [51]

The second partially-solved sub-problem is efficient subsequence–visual feature mapping. Dotplot has quadric subsequence-visual feature mapping. In other words, *n* repeated symbols give rise to n^2 corresponding visual features. Thus, 70% is deducted from the *dos*, while 30% remains, only due to the symmetric beauty generated by this mapping.

There are no conflict methods. The *DoS* of Dotplot is as follows:

$$DoS = \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k}$$

$$= \frac{5*1.0+5*0.6+5*1.0+4*0.3+3*1.0+5*0+4*0+2*0}{5+5+5+4+3+5+4+2}$$

$$\approx 52.1\%$$

Equation 2.3. *Dos* of Dotplot for *TP*

2.5 Step Five: Improve the Temporary Infovis Technique

The previous step points out the possible areas for improvement. Specific improvements involve various unpredictable situations and thus are out of the scope of the discussion in this dissertation.

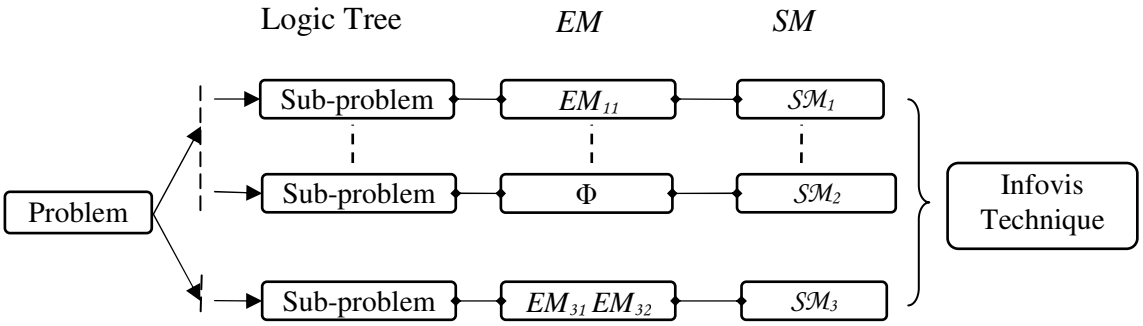


Figure 2.9. Improve a Temporary Infovis Technique

Figure 2.9 illustrates the improvement of a temporary infovis technique. Similar to the expression of a temporary infovis technique, the improved infovis technique is a combination of solution methods (denoted as *SM*) and can be represented as follows:

$$\text{Infovis Technique} = \{SM_i \ldots SM_x\}$$

Equation 2.4. Infovis Technique Expression

2.5.1 Example

Wattenberg [126] invented the arc diagram, which is capable of showing complex patterns of repetition in strings. An arc diagram extends the idea of musical AABB notation by utilizing an arc representing consecutive repeated sub-strings. The original arc diagram proposed by Wattenberg only provides static graphical representation, and no dynamic exploration is supported. Figure 2.10 illustrates an enhanced arc diagram in which simple interactions are supported. The section below explains how the enhanced arc diagram solves the *TP*: visualizing structures in strings.

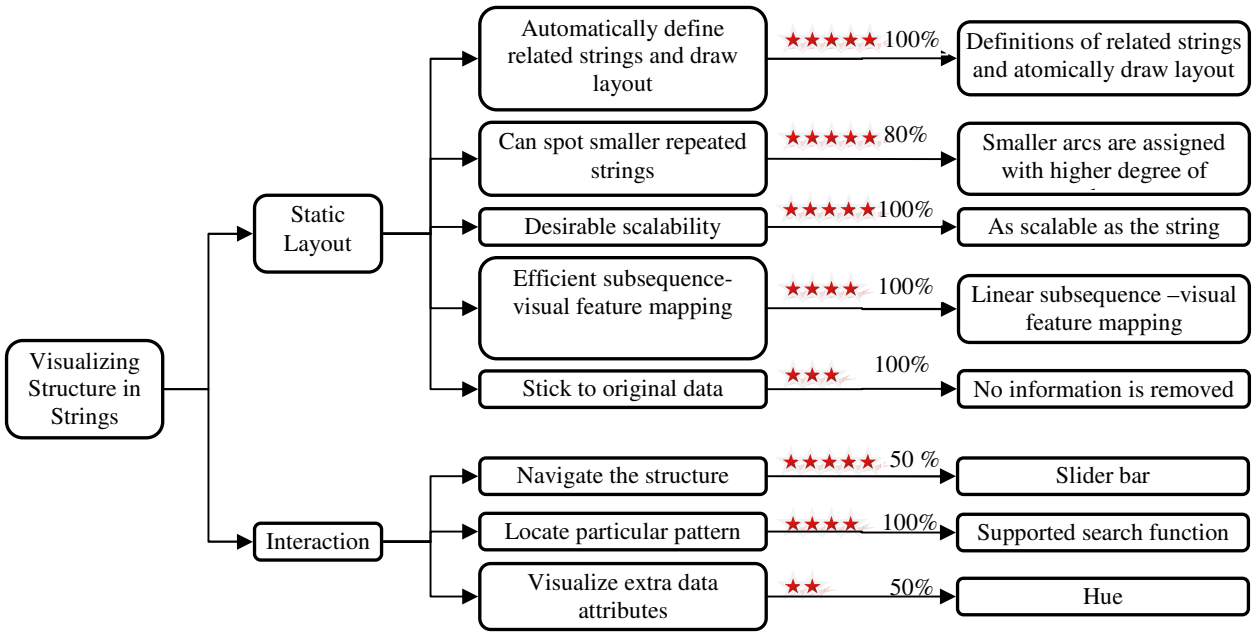


Figure 2.10. Evaluation of Enhanced Arc Diagram

Static Layout:

- Automatically define related strings and draw layout.

  An arc diagram is built based on the idea of showing only essential substrings, or those that are crucial to a string's structure. Three definitions clarify the meaning of essential substrings. Any two consecutive repeated essential substrings are automatically identified by a pattern-matching algorithm and are linked by a translucent arc.

- Spot smaller repeated strings.

  An arc diagram overlays arcs with different degrees of translucency, which usually increases when the level of an arc decreases. This method highlights smaller repeated strings while still keeping the larger repeated strings as context, and therefore no match is completely obscured. However, if a string's structure is very complex and too many levels are shown, as illustrated in Figure 2.11, the generated arc diagram inevitably becomes too cluttered and is therefore uninformative. This clutter can be avoided by restricting the number of levels shown. Because of the potential for clutter, 20% is deducted from *dos*.

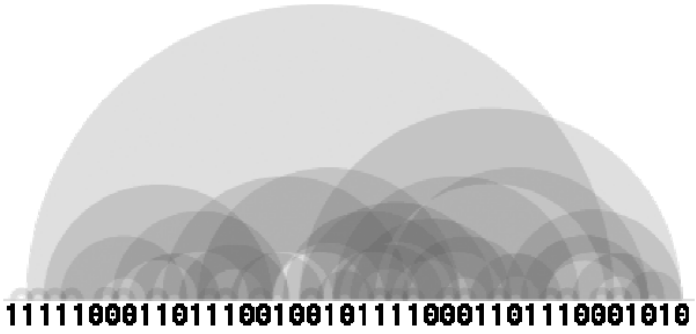


Figure 2.11. An arc diagram with too much detail [126]

- Desirable scalability.

  Unlike Dotplot, an arc diagram only connects consecutive sub-strings rather than every pair of repeated ones. This allows an arc diagram more desirable scalability than Dotplot.

Moreover, an arc diagram requires a much smaller display area than Dotplot, making an arc diagram more adaptive to large data sets. Theoretically, an arc diagram is as scalable as the underlying data. Arc diagrams can be utilized to visualize a string's structure whenever the string can be written in text on screen.

- Efficient subsequence-visual feature mapping.

  When a subsequence repeats *n* times, only *n-1* corresponding arcs are highlighted in an arc diagram. This linear mapping is more efficient than the quadric mapping of Dotplot.

- Stick to the original data.

  An arc diagram identifies and connects consecutive repeated sub-strings strictly according to the original data's ordering. No attributes of the original data are removed.

Filter Interaction:

- Navigate the structure.

  A slide bar controlling the level of details helps users drill down in the enhanced arc diagram. However, it cannot perfectly solve this sub-problem because it only allows users to vertically explore the structure. Due to the slide bar's inability to navigate horizontally, 50% is deducted from *dos*.

- Locate specific patterns.

  A search function is supported to find specific patterns. The resulting patterns are highlighted once found.

- Visualize extra data attributes.

  Visual feature hue is used to denote an additional attribute of the data (for example, volume in a melody). While using hue is a common method for illustrating distribution, overlaid arcs

cause different hues to mix, which easily leads to misunderstanding. Because of the potential for hue mixture and misunderstanding, 50% is deducted from *dos*.

The *DoS* of enhanced arc diagram is:

$$DoS = \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k}$$

$$= \frac{5*1.0+5*0.8+5*1.0+4*1.0+3*1.0+5*0.5+4*1.0+2*0.5}{5+5+5+4+3+5+4+2}$$

$$\approx 86.3\%$$

Equation 2.5. *Dos* of Enhanced Arc Diagram

This section introduces each step in 5DITS, but not every step is necessary. Some of them can be omitted due to the requirements of real situation. Take the InfoShape for example; the triggering problem is so specific that there is no need to break it down to smaller problems. Then, the next three sections apply 5DITS to three different triggering problems, respectively.

CHAPTER 3

RELT: VISUALIZE HIERARCHICAL INFORMATION ON SMALL SCREENS

3.1 Introduction

The past few decades have witnessed an information explosion. Visual tools such as charts, diagrams and maps assist people to identify trends, problems, locations and navigate vast amounts of information. With the increasing processing power of computers and rendering capabilities of display technology, sophisticated infovis techniques have been widely applied to various application domains. Numerous kinds of application data, such as financial statistics, remote-sensing results, real-time data, gene sequences, can be intuitively represented and analyzed using infovis tools.

Many types of modern-day information are organized in hierarchical structure. Typical hierarchical data contain product catalogs, web documents, computer file systems, organizational charts, and so on. Visualization of hierarchical data not only improves the understanding of the data and the relationship among the data, but also enhances the way of manipulation and navigation.

"Hierarchies are one of the most commonly used structures…

Hierarchical structures not only plan significant roles in their own right, but also provide a means of representing a complex structure in a simplified form. "

– Chen [25]

A hierarchy, usually termed a tree, is a connected acyclic graph $T= (N, E)$. N denotes the set of nodes and E denotes the set of edges. A rooted tree $T= (N, E, r)$ is a tree T having a distinguished

vertex *r* as root. The *depth (level)* of a node denotes the number of steps from the root to it along the path. If two nodes are linked by an edge, the one with smaller depth is the parent, and the other node is the child. A node may have several children but only have one parent. Nodes with the same parent are called siblings. Nodes are usually divided into two types: inner node having at least on child and leaf node having no children.

In addition, research [1] has been conducted on transforming a graph to a tree representation. The ability of visualizing and navigating hierarchical information is urgently needed and becomes an emerging challenging research topic. Various approaches have been investigated for visualizing hierarchies and can be generally classified into connection and space-filling categories.

Connection approaches draw a hierarchy in a node-link diagram. Each piece of information is visualized as a node. An edge depicts the relationship between two connected nodes. Space-filling approaches use position and adjacency rather than links to express the rational relationship. Avoiding using explicit edges increases the utilization of display area.

Both the connection and the space-filling approaches are effective methods to visualize hierarchies. Although the effectiveness varies according to the different situation and applications, connection approaches are essentially advantageous in revealing the information structure while space-filling approaches are good at maximizing the space usage.

With the increasing number of users of mobile devices, especially considering the soaring number of smart phones, the need for searching and navigating data on small screens becomes apparent. Various visual approaches have been proposed for browsing the web information on small screen devices 0 [13]. Some of these approaches involve innovative interaction techniques [33] [130]. Although many infovis techniques have been proposed for viewing tree structures on

big screens, little progress has been made for displaying trees on small screens. Effective and efficient navigation through hierarchical structures on limited viewing spaces has not been widely investigated. The next section reviews several classic hierarchy visualization techniques designed for big screen and their adapted versions for small screens.

3.2 Related Work

3.2.1 Connection Approaches

Connection approaches map a hierarchy into a node-link diagram. Each piece of information is visualized as a node whose visual features such as color and shape depend on various applications. A set of explicit edges are between parents and their children. Connection approaches naturally match the human perception of hierarchy, so the graphical representations generated by connection approaches usually have the advantage of conveying relationships and structure of underlying data.

All the connection approaches essentially focus on two major layout issues:

How to locate nodes and how to connect nodes?

Typical connection approaches include: classic tree drawing [103] [124], tree browser, radial tree [38] [53] [54] [7], balloon views [53] [63] [84] [121], bubble tree [16], hyperbolic tree[75] [86] [141] [142], cone trees [53] [107] [108], magic eye view [19] [7], botanical tree [71].

3.2.1.1 Classic Tree Drawing

The classic tree drawing shown in Figure 3.1 by Reingold and Tilford [103] is possibly the best known hierarchy visualization technique. The Reingold-Tilford algorithm first calculates the nodes' positions. The tree is drawn recursively upward from bottom to top. Leaves are positioned in layers based on their depth; their horizontal positions are arbitrarily given to keep the nodes in

the same layer as flush as possible. Each sub-tree is drawn independently no matter where it appears in the tree. All parent nodes are forced to be placed centrally above their children. When nodes of a sub-tree are located, the sub-tree is added edges and is moved to its adjacent sub-tree as close as possible. The Reingold-Tilford algorithm is fast, predictable and achieves an "esthetically pleasing" [103] layout. Walker [124] and Buchheim [18] later proposed an improved algorithm which can create left-to-right and right-to-left layouts.

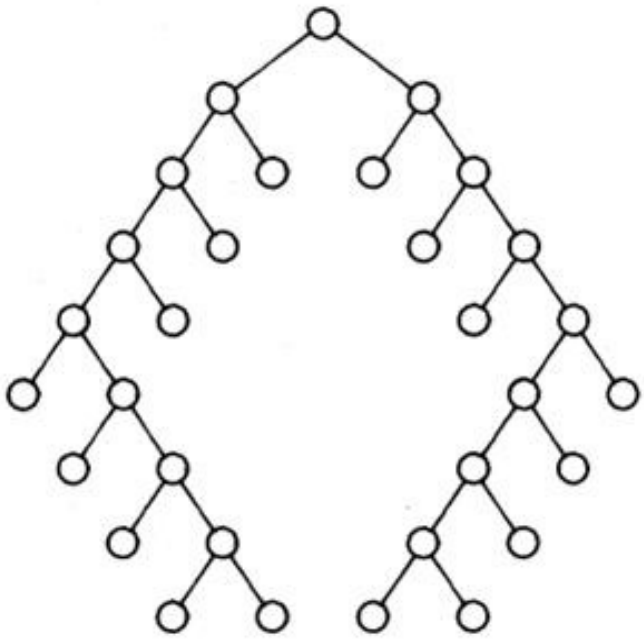


Figure 3.1. Reingold and Tilford Layout [103]

3.2.1.2 Tree Browser

A tree browser utilizes the outline method to visualize hierarchies and is widely adopted on computers by millions of people. Figure 3.2 shows windows explorer, a very classic tree browser. A tree browser usually has two panels. The left-hand panel visualizes a hierarchy structure and the right-hand one lists the children of selected items.

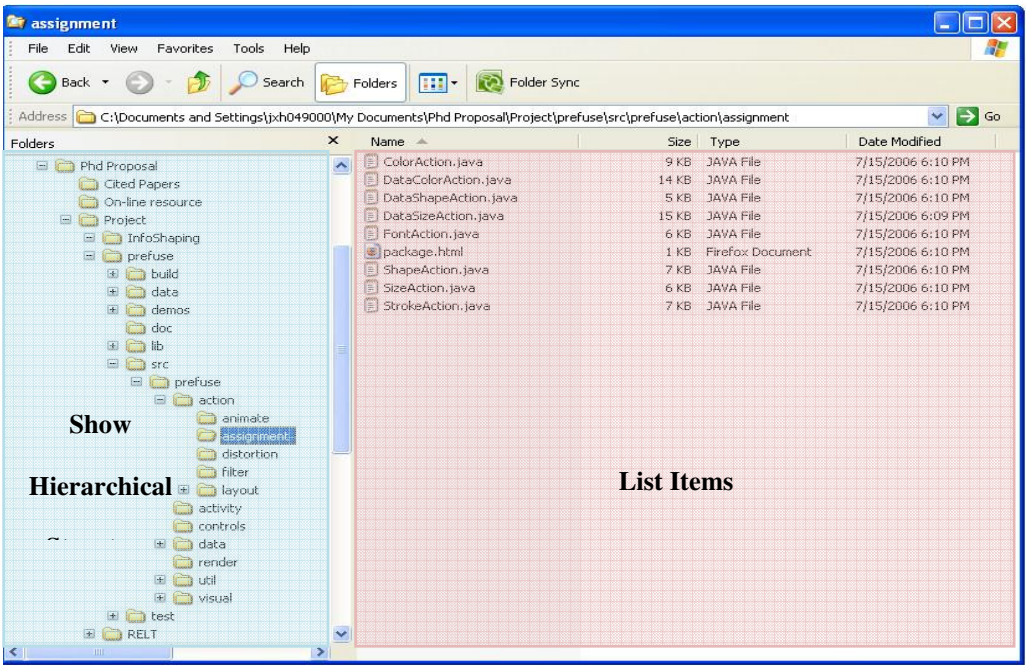


Figure 3.2. An Example of Tree Browser: Windows Explorer

Only some default sub-trees of the visualized hierarchy are expanded when the tree browser is initially opened. By expanding and collapsing the sub-trees, users have the biggest freedom to navigate a hierarchy. One obvious disadvantage of a tree browser is its undesirable scalability. Because the expended nodes of the hierarchy are vertically located on the left panel, a scrollbar is required. When the hierarchy is large, especially many subtrees are expanded; users may have too much scrolling.

3.2.1.3 Radial Tree

Eades proposes a radial tree [38] [53] algorithm which places nodes on concentric circles based on the nodes' depth. As Figure 3.3 illustrates, this algorithm recursively locates the children of a sub-tree into circular wedges. The spanning angle of a circular wedge is proportional to an attribute of the node, such as weight. Wedges in Figure 3.3 are outlined for two drawn nodes in the first and second level. Each node has its own wedge where the sub-tree rooted from the node

is located. The elegance of this technique is that it keeps adjacent branches from overlapping. A radial tree is simple, predictable and performs well, especially for small and compact trees. However, two disadvantages prohibit this technique from being applied widely. The first is non-economic screen estate. The algorithm cannot avoid wasting a large portion of screen. Second, this algorithm behaves undesirably for large trees, because the nodes at high level can hardly be distinguished.

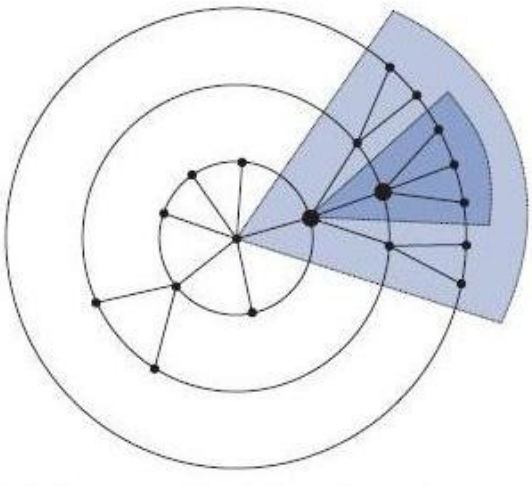
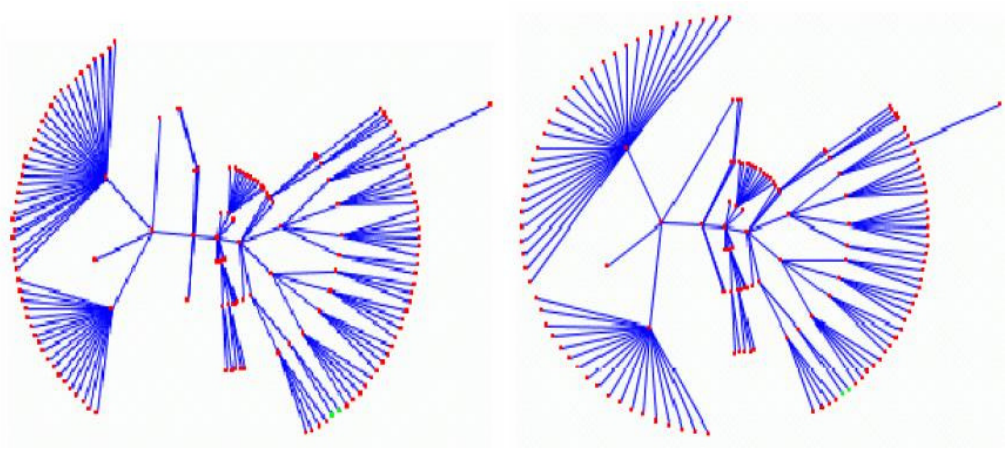


Figure 3.3. A Schematic Illustration of a Radial Tree. [7].

Herman *et al*. [54] improved screen usage by dropping the convexity constrain property (the original algorithm forces all wedges to keep convex so that no wedge is assigned an extreme big angle). Layouts in Figure 3.4 (a) and (b) are generated by Herman's algorithm and original algorithm, respectively. Clearly, Herman's algorithm utilizes more space; however, it introduces intersections when it is applied to a big tree.

Regarding the original algorithm's undesirable application on a large hierarchy, Jankun-Kelly and Ma [61] presented the MoireGraph. Focus plus context technique is integrated into the MoireGraph. The focus node is located at the center of circle and its children are also located on

relative concentric circles. The siblings of the focus node are also drawn on the concentric circles with smaller wedges.



(a) Radial tree with Convexity Check

(b) Radial Tree without Convexity Check and Statistical Modification

Figure 3.4. Radial Tree Layouts. [54].

3.2.1.4 Balloon Tree

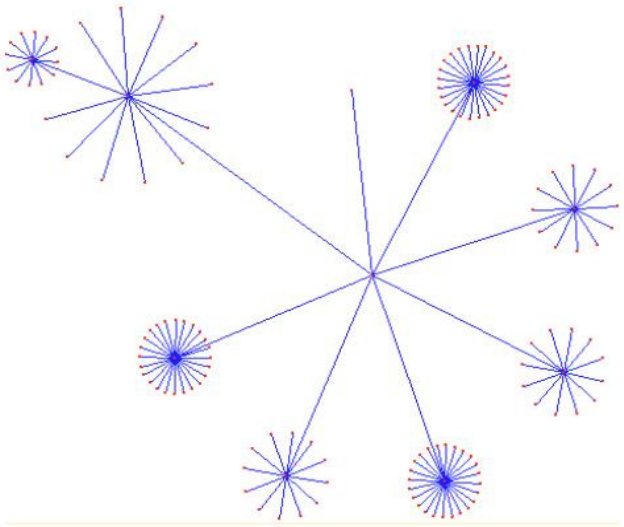


Figure 3.5. Balloon View of a Hierarchy. (Circular Tree Layout) [84].

Teoh and Ma introduced "RINGS" [121] which shows maximum contextual information and retains the clarity of the focus area. As illustrated in Figure 3.6, RINGS positions each node as a

circle in concentric rings around the center of the parent circle. A node and its children are located in the circle which is assigned to the node. The size of the circle is decided by the number of children. The bigger circle is positioned relatively outer than the smaller circle. Edges are colored according to their distance to the root. Although the author claims that RING is to "make more efficient use of limited display space," this circular layout essentially inevitably wastes the display area.

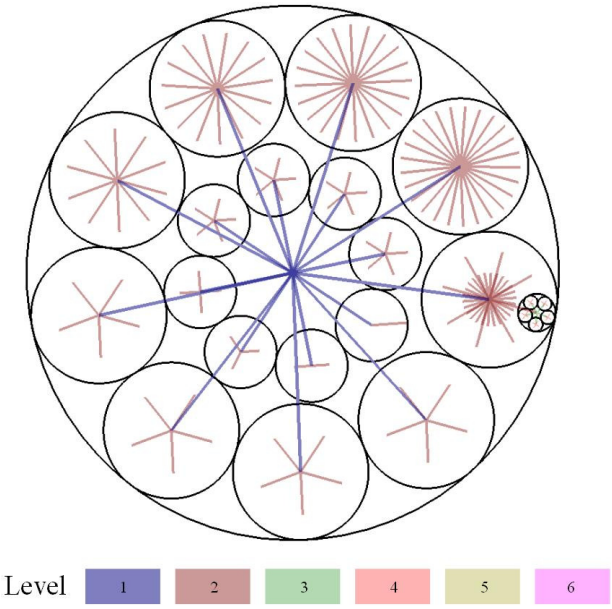


Figure 3.6. RINGS [121].

Balloon view in Figure 3.5, which is also called circular tree, was introduced by Melancon and Herman [84]. This hierarchical layout looks similar to the radial tree. Instead of locating nodes on concentric circles, balloon view assigns a circle to each inner node. The algorithm positions an inner node in the center of its associating circle, and the sub-tree which is rooted at this node is located on the circumference of this circle. The higher level an inner node has the smaller radii the node's circle has. Users can select a node to switch its interest. After a node is selected, the algorithm makes the selected node as the root and then reorganizes the whole tree. Similar to the

radial tree, balloon view is not suitable for a large tree. Moreover, this visualization method does not economically utilize the display area.

3.2.1.5 Bubble Tree

Bubble tree introduced by Boardman [16] looks similar to a balloon tree. The algorithm recursively sub-categorizes a tree into sub-trees according to the tree's natural structure. Each sub-tree is visualized as a bubble which contains sub-tree's root and its children. Initially, only the opaque root bubble is visible. Boardman provides "bursting bubbles" [16] to navigate a hierarchy. The "bursting bubbles" contains three detail increasing actions.

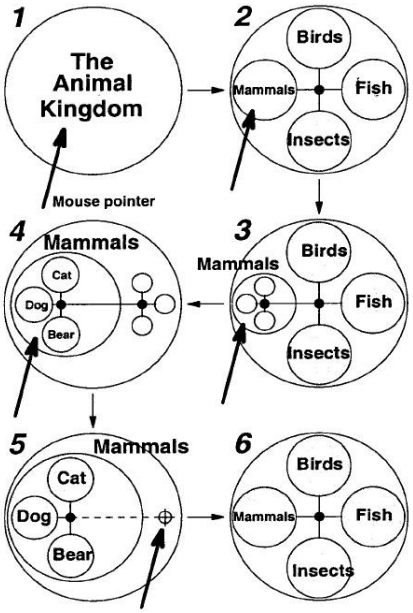


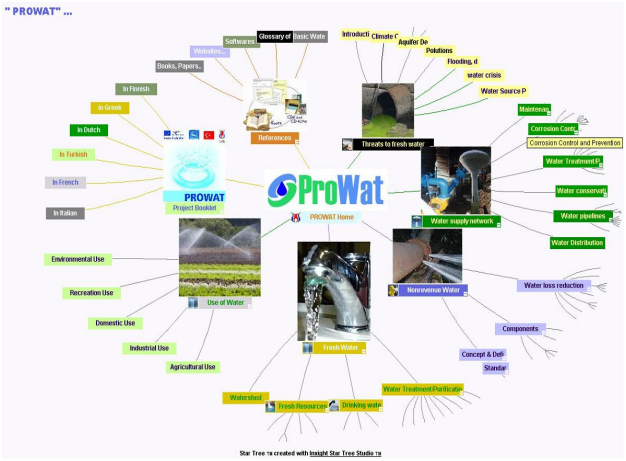
Figure 3.7. Bubble Tree [16].

The first detail increasing action reveals the children of a bubble by a left click. First detail increasing action drives frame 1 in Figure 3.7 to frame 2 and then to frame 3. Children bubbles, which are inner nodes, also can be burst by the first detail increasing action. The second detail increasing action, which leads frame 3 to frame 4, and then to frame 5, not only expands the selected nodes, but also shrinks the other bubbles' size. A focus is the third detail increasing

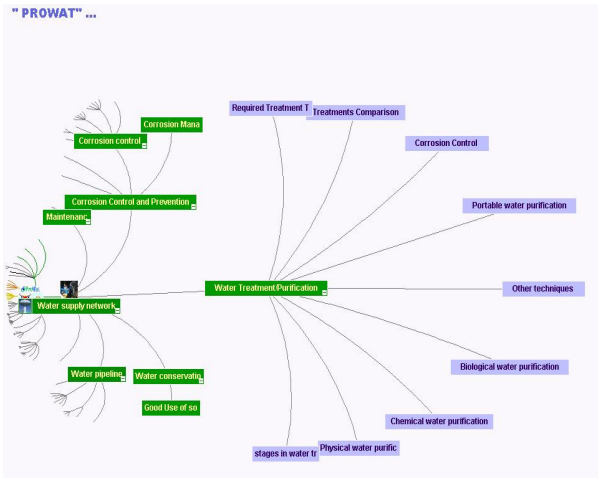
action, which the author regards as “the most powerful of the three detail increasing actions.” A focus in the bubble tree expands the selected node and moves it to the center, while other bubbles become opaque. Frame 6 illustrates the result of a focus in frame 5. The Mammals node is expanded and moved into the center of the circle; all other nodes become invisible.

3.2.1.6 Hyperbolic Browser

Hyperbolic browser, introduced by Lamping *et al.* [75], is one of the most well-known commercial hierarchy visualization techniques. A hyperbolic layout is generated by mapping a hierarchy in hyperbolic geometry to Euclidean space.



(a)



(b)

Figure 3.8. Hyperbolic Browser Implemented by Inxight Star Tree Studio TM [142]

First, a hierarchy is laid out radially as a node-link diagram with the root in the center on a hyperbolic plane. Because the hyperbolic plane has unlimited space in each direction, the entire tree can always be drawn on the plane. Each node on the hyperbolic plane allocates nearly the same amount of space to its children regardless of how deep the children are in the tree. Second, a mapping transforms the hierarchical layout to a unit disc [75] or a unit sphere [86] in the Euclidean space. The transformed hierarchical layout on the unit disc no more remains the approximate same density. Then density grows towards to the border of the disk. This desirable feature naturally realizes the focus plus context function. The node in the center tends to have more space and therefore can be observed clearly. The nodes crowded near the border provide context information of the tree. Users can drag the tree around to browse it.

The algorithm also allows users to select a focus point which can automatically move to the center. A smooth animation is implemented to keep the user's mental map. The hyperbolic browser shown in Figure 3.8 is a demo [141] implemented by Inxight Star Studio [142]. Figure 3.8 (b) shows the result of dragging a new node into the center.

Hyperbolic browser provides focus plus context navigation and is capable of navigating large a hierarchy. One disadvantage is that the geometrical background is sometimes hard to understand.

3.2.1.7 Magic Eye View

Another focus plus context hierarchy visualization technique is magic eye (shown in Figure 3.9) which was proposed by Bürger [19]. Instead of constructing a layout using hyper geometry in hyperbolic view, the magic eye browser constructs the layout based on spherical projection.

Magic eye takes two steps to construct the layout. In the first step, a radial hierarchical layout which is generated by Walker algorithm [124] is mapped to the surface of a hemisphere. The mapping function uses polar circle length of nodes as spherical angle. The second phase projects

generated in the first step onto the equatorial circle. A special point termed projection center lies on the equatorial circle and determines the focus area. The tree can be moved on the hemisphere. If the projection center does not lie in the original point, the hierarchical layout is distorted. This technique produces a layout where nodes on a concentric circle have different radii.

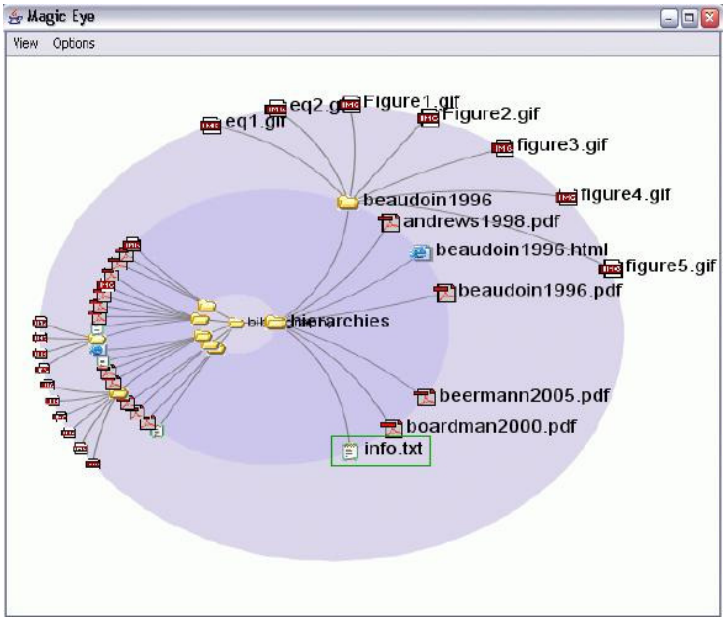


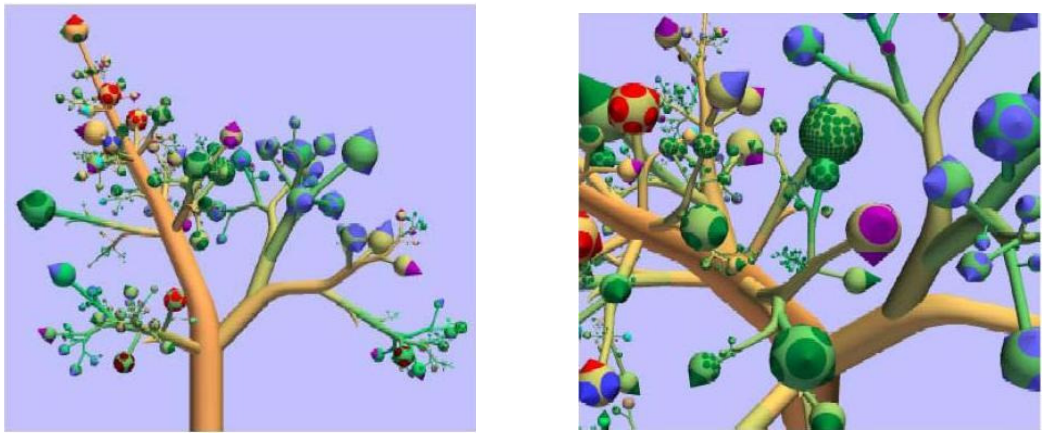
Figure 3.9. Magic Eye [7].

In contrast to moving the tree in the hyperbolic browser, users move the projection center to browse the tree in the magic eye browser. A sub-tree is enlarged when the projection center is moved to it. The magic eye technique is suitable for a small or middle sized hierarchy. Unlike the infinite space in hyperbolic sphere, the size of hemisphere is constant; therefore, magic eye is not suitable for a large hierarchy.

3.2.1.8 Botanical Visualization

Botanical visualization, proposed by Kleiberg *et al*. [71], uses the botanical tree metaphor to visualize a hierarchy in three-dimensional space. This technique is based on the fact that people can easily perceive branches, leaves and their arrangement in a large botanical tree (shown in

Figure 3.10). Inner nodes are visualized as branches and the leaves are represented as fruits on branches. To achieve better esthetics, continuing branches are emphasized and long branches are contracted. Botanical visualization only supports the zooming technique during navigation. Users may easily get lost due to the lost context during navigation.



(a) Overview of a Botanical Tree (b) Zoom in a Botanical Tree to Navigate

Figure 3.10. Botanical Visualization. [71].

3.2.2 Space-filling Approaches

Node-link diagrams constructed by the aforementioned connection approaches cannot avoid wasting screen due to explicit edges. Instead of using explicit edges, space-filling techniques use the position and adjacency to express the parent-child relationship. By getting rid of edges, space-filling techniques have the advantage of utilizing maximum display area. Following sections introduce some classic space-filling hierarchy visualization techniques.

3.2.2.1 Treemap

Treemap, which was designed by Shneiderman [113], is utilized as a commercial tool in many domains. In contrast to node-link diagrams, Treemap divides the screen into nested vertical and

horizontal areas. Each area expresses a node, and enclosure relationships express the parent-children relationships. The size of an area is proportional to the weight of a node.

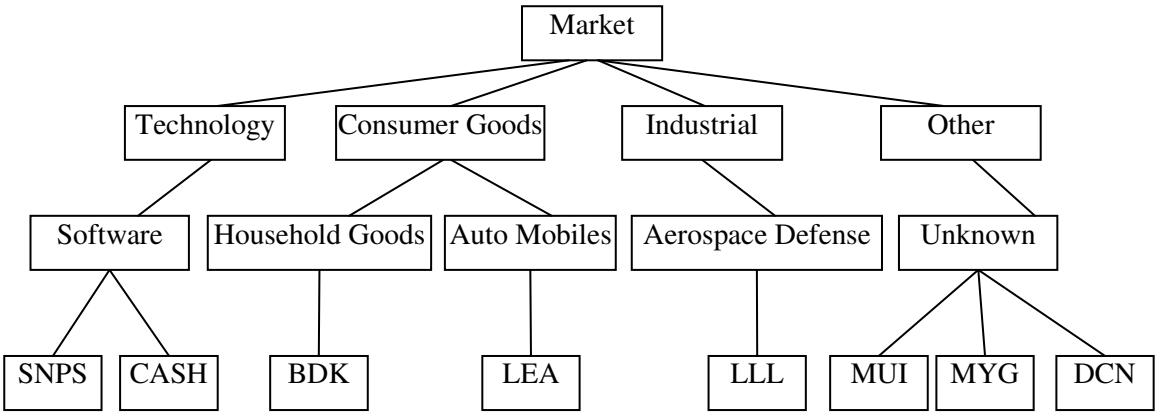


Figure 3.11. A Typical Stock Market Classification

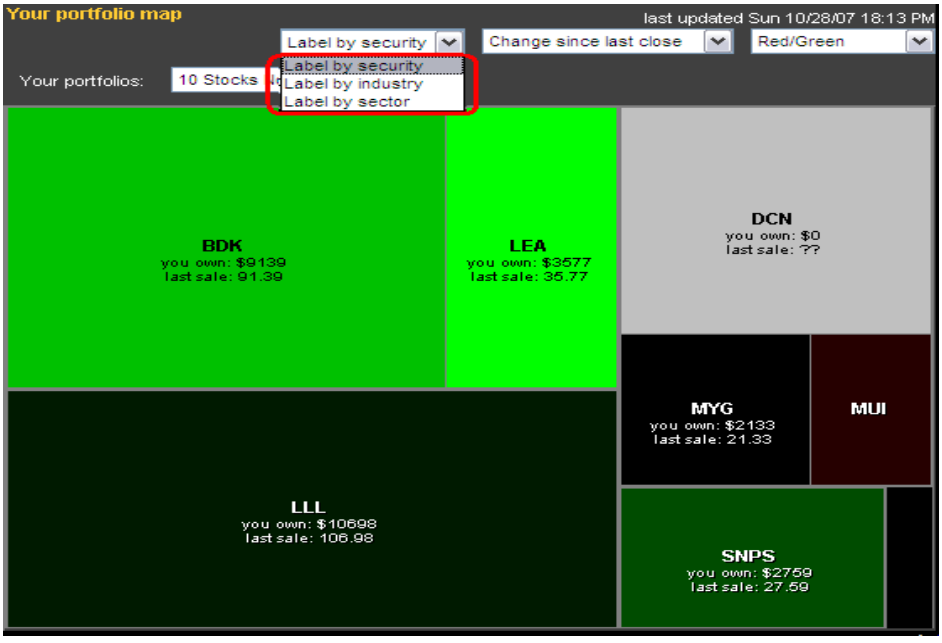


Figure 3.12. Visualization of Treemap for Stock Market Visualization [150]

Figure 3.12 illustrates a Treemap tool called portfolio map [150] for the stock market shown in Figure 3.11 which is “10 stocks now 799”. Traditional stock coloring indicates stock price performance. Green stands for price up and red for down, black indicates an unchanged price,

and gray means that users do not own any shares. The rectangle size indicates the market capitalization of the corresponding company.

Treemap has the prominent advantage of 100 percent display area usage. However, hierarchical structure which is expressed by enclosure is not as clear as the one which is illustrated by edges. In addition, the algorithm of Treemap only fits the rectangular display area.

3.2.2.2 Voronoi Treemap

Previously discussed Treemap algorithm is only limited to a rectangular display area. Balzer and Deussen [11] introduced the voronoi Treemap which eliminates this problem through enabling subdivisions of and in polygons. The voronoi Treemap algorithm is constructed based on the centroidal voronoi tessellations [37], which are widely used to minimize the energy or cost in many domains. Figure 3.13 [11] illustrates the voronoi Treemap for a 10 level hierarchy with 4075 nodes. Color brightness is used to encode the levels; a brighter color denotes a lower hierarchy level.

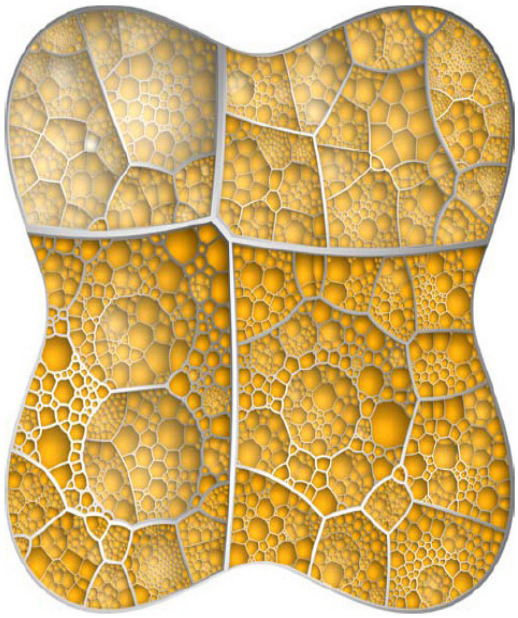


Figure 3.13. Voronoi Treemap [11].

3.2.2.3 Information Slices

Information slices was introduced by Andrew and Heidegger [6]. This technique utilizes one or more 2D semi-discs to visualize hierarchies. Each disc expresses multiple levels of a hierarchy and the number of levels shown in each semi-disc is user configurable. A series of discs can be cascaded to visualize the deeper hierarchies. At each level of a semi-disk, the children are fanned out according to the total size of the child. Information slices performs well especially for deep hierarchies.

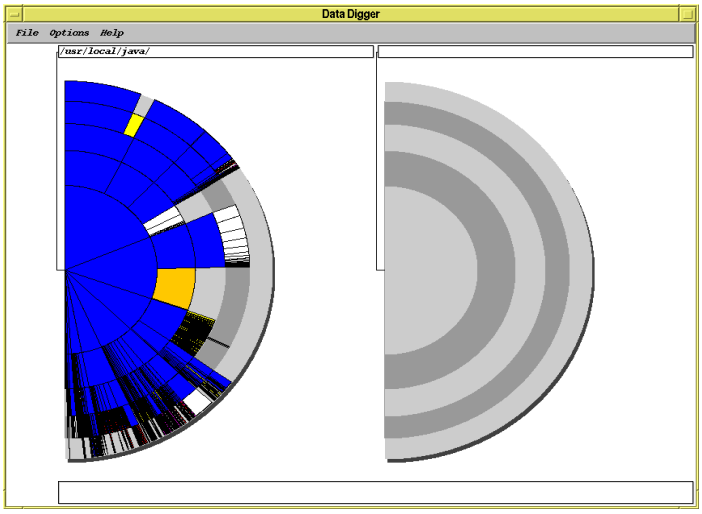


Figure 3.14. Information Slices Initial Layout of JDK 1.1.6 Distribution [6].

Andrew and Heidegger [6] introduced a prototype of information slices to visualize the hierarchical structure of Sun Microsystem's JDK 1.1.1 distribution for Solaris. This file system consists of 6158 files arranged in 502 directories. Each disk is initially set to visualize five levels of hierarchy. The left semi-disk visualizes the root directory and its five levels subdirectories. Directories are colored blue and files are colored according to their types. The right-hand semi-disk is not used. Figure 3.15 illustrates the display when a user expands the swing/com/sun/java/swing subdirectory in the fifth level into the right-hand disc.

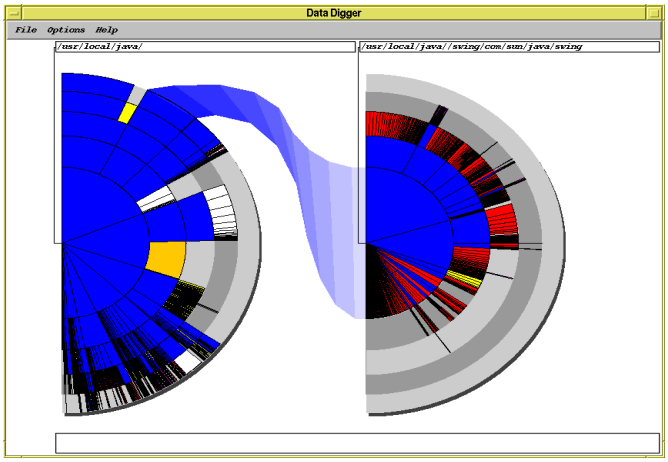


Figure 3.15. Information Slices of JDK 1.1.6 Distribution [6].

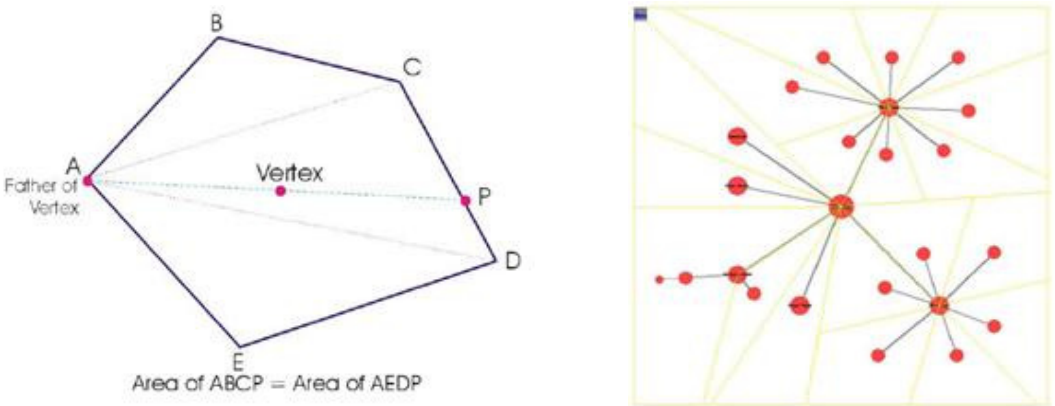
3.2.3 A Space-Optimized Tree Visualization

Although previous discussed connection approaches have been proposed for handling hierarchy visualization, few of them consider the issue of optimizing the usage of display area. Space-filling techniques, on the other hand, indeed show their advantages of utilizing display area. However, the generated relational structure is not clear due to the lack of explicit edges.

Vinh and Maolin [87] proposed a space-optimized technique combining the advantages of both connection approaches and space-filling approaches. This technique optimizes the node-link diagrams of trees and increases the usage of the screen by adding more nodes on a limited space. Unlike other connection approaches that directly draw node-link diagrams, this technique first divides the whole display area into a set of geometrical areas which are used to define sub-tree layout. Figure 3.16 illustrates an example of area division. The area division ensures 100% space utilization. After that, each node is assigned and positioned on an area. Figure 3.16 (a) illustrates how a vertex v (node) is positioned on its assigned area. First, a point P should be found on the boundary that the line connecting P and the parent of v evenly divides the area (Size of ABCP

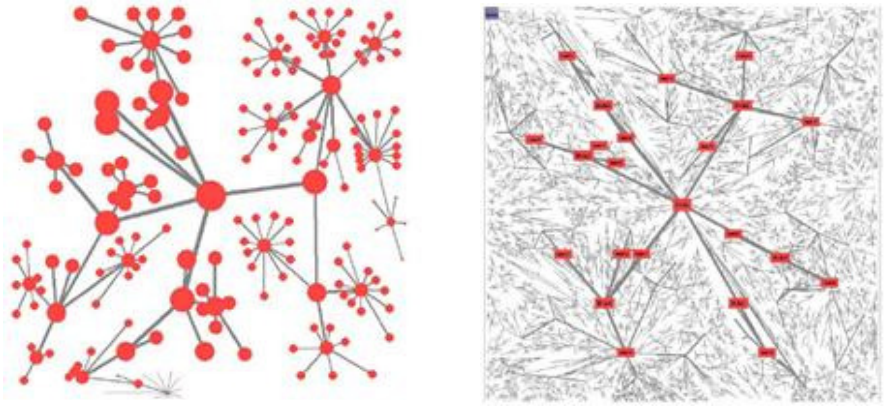
equals the size of APDE). Vertex *v* is then positioned in the midpoint of the line. Figure 3.16 (b) illustrates an example of area division and vertex position.

Space-optimized tree visualization combines the advantages of both connection approaches and space-filling approaches. Its ability of maximally using display area enables it to be well scalable to a huge hierarchy.



(a) (b)

Figure 3.16. Example of Vertex Position and Area Division [87]



(a) A Medium Large Data Set of (b) A Date Set of Approximately 9000 Nodes

Figure 3.17. Examples of Space Optimized Tree [87]

Figure 3.17 shows example screen dumps of using the space-optimized visualization technique on two data sets with different size. The screen has a standard resolution (1024 x 768).

Figure 3.17 (a) shows the layout of a medium data set consisting of about 150 nodes.

Figure 3.17 (b) shows visualization on a data set which has nearly 9000 nodes.

3.3 Applying 5DITS to Design RELT

Mobile devices, especially cell phones, have been widely accepted as an important part of peoples' daily lives. The technique breakthrough in the communication area, mobile computing area, and human-computer interaction area changes the way we think, work, live and even love. All these technology advancements not only offer people unprecedented mobility, but also provide mobile devices the power of dealing with work which used to be handled on PCs or laptops. The seemingly ever-increasing smart phone market is a sign of this trend. A smart phone usually runs a complete operating system, and therefore provides PC-like functionalities. Demands for appropriate infovis techniques come with the need of dealing with data on mobile devices. Most current infovis techniques are simply adapted from the version used on PCs, rather than especially for mobile devices. This section applies 5DITS to design an infovis technique especially for viewing hierarchies on mobile devices.

3.3.1 Define Triggering Problem

Table 3.1. Comparison between Mobile Devices and PCs

Feature	Mobile Device	Desktop
Screen Size	Small	Large
Colors	Few	Many
Input Devices	Button, Pen, Finger	Mouse Keyboard
Width/Height	Varied	Around 4/3
Connectivity	Slow	Fast

With the increasing number of users of mobile devices, such as iphones and android phones, the need for searching and navigating data on small screens becomes apparent. Various approaches

have been proposed for browsing the web information on small screen devices [9] [13] and some with innovative interaction techniques [33] [131]. Researchers have found that simply scaling infovis methods designed for desktops to fit mobile devices would not be a good choice. This is because of the essential limitations in downward scaling, as compared in Table 3.1.

Although many infovis techniques have been proposed for viewing hierarchical structures, little progress has been made for displaying hierarchies on small screens. Effective and efficient navigation through a hierarchy on limited viewing spaces has not been widely investigated. Compared to other feature gaps which are becoming smaller and smaller, the gap of screen size between mobile device and desktops can hardly be changed. The limited size of a mobile screen is the crucial thing considered during the design process of hierarchy visualization on a mobile device. Therefore, the triggering problem is defined as "Visualize Hierarchical Information on Small Screens".

3.3.2 Decompose Triggering Problem

Although people tend to view data on their mobile devices, their behaviors and operations on mobile devices, even for exactly the same task, differ from the ones on PCs. The most salient trait is that people prefer to perform easier tasks on mobile devices, and their needs on mobile are usually not as complex as ones on PCs. According to the relatively simple needs, we decompose *TP* into three sub-problems, as illustrated in Figure 3.18.

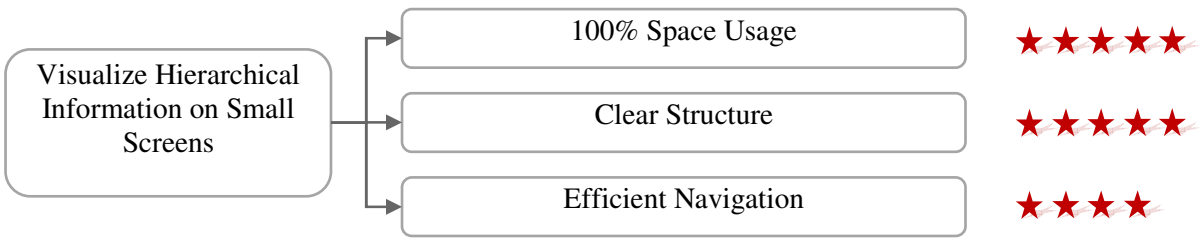


Figure 3.18. An Example Logic Tree

- 100% Space Usage:

As per previous discussion, a limited display area is the most valuable resource on mobile devices. We have to squeeze maximum utilization out of every pixel on the screen. Instead of saying "maximally increase the screen usage," which can rarely be quantitatively measured, we make it much stricter to "100% Space Usage." This requires that the expected infovis technique produces a layout where each pixel is used to denote either a part of an edge or a part of a node. The goal is that no blank space exists and every pixel tells.

- Clear Structure:

Although some space-filling approaches, such as Treemap, can achieve 100% screen usage, they do not express hierarchical structure as clear as node-link diagrams. First, the parent-child relationship, usually denoted by enclosure, is less explicitly shown as one in node-link diagrams. Second, most space-filling approaches toss out the ordering information of siblings. This keeps space-filling approaches from the situation where the order of sibling matters. We believe structural clarity is as important as 100% screen usage. Sacrificing structural clarity to achieve desirable screen usage is "rob Peter to pay Paul" and thus is unacceptable. In our scenario, structure clarity does not associate with personal estheticism, but only considers two issues: clear parent-children relationship and clear sibling-sibling relationship.

- Efficient Navigation

An efficient navigation usually has a clear hint of static layout and supportive interaction. A good hint of static layout clues users where they are when they are exploring a part of a hierarchy. In additional, a useful hint should bring a user's attention to the currently reviewed node. Good supportive interaction aims at assisting users to go through a hierarchy, saving time and clicks.

3.3.3 Design a Temporary Infovis Technique

We discussed node-link layouts and space-filling layouts in Section 3.2. Inevitably, both layouts have their disadvantages.

From the structural clarity point of view, the best things in node-link diagrams are explicit edges which clearly denote the rational relationships. However, if we switch to perspective on screen usage, the worst things in node-link diagrams are also edges which occupy display area extremely inefficiently. Figure 3.19 illustrates big portions of unused space in previously discussed node-link diagrams. Achieving structural clarity and maximally using display area are essentially conflicts for node-link diagrams. Even for space-optimized tree visualization [87], the collection of areas which is assigned to nodes fully covers the whole display area; however, the visual nodes do not. No node-link diagram can avoid structure clarity – maximum screen usage conflict. In addition, nodes in node-link diagrams are usually too small on a small screen; they cannot be selected as easily as when they are on a big screen. This increases the difficulty of navigation.

Similarly, a structural clarity–maximum screen usage conflict remains in space-filling techniques. Display space is maximally saved by getting rid of visible edges. Rational structure is represented by enclosure which does not naturally fit people's cognition of a tree. People need additional cognitive effort to understand a hierarchy's structure as the cost of efficient space usage. There are two problems for layout generated by space-filling approaches during navigation. Take Treemap for instance; the first problem is that children are not visually sorted. If users want to view all children which are visually represented as small rectangles enclosed in a bigger rectangle denoting their parent one by one, they will see small rectangles one after another in a seemingly ruleless manner. The second problem appears when Treemap is used on mobile

devices such Iphone, which uses tough screens. Because when a user selects a node, it's hard to know if the user wants to select the associated leaf node or its parent.

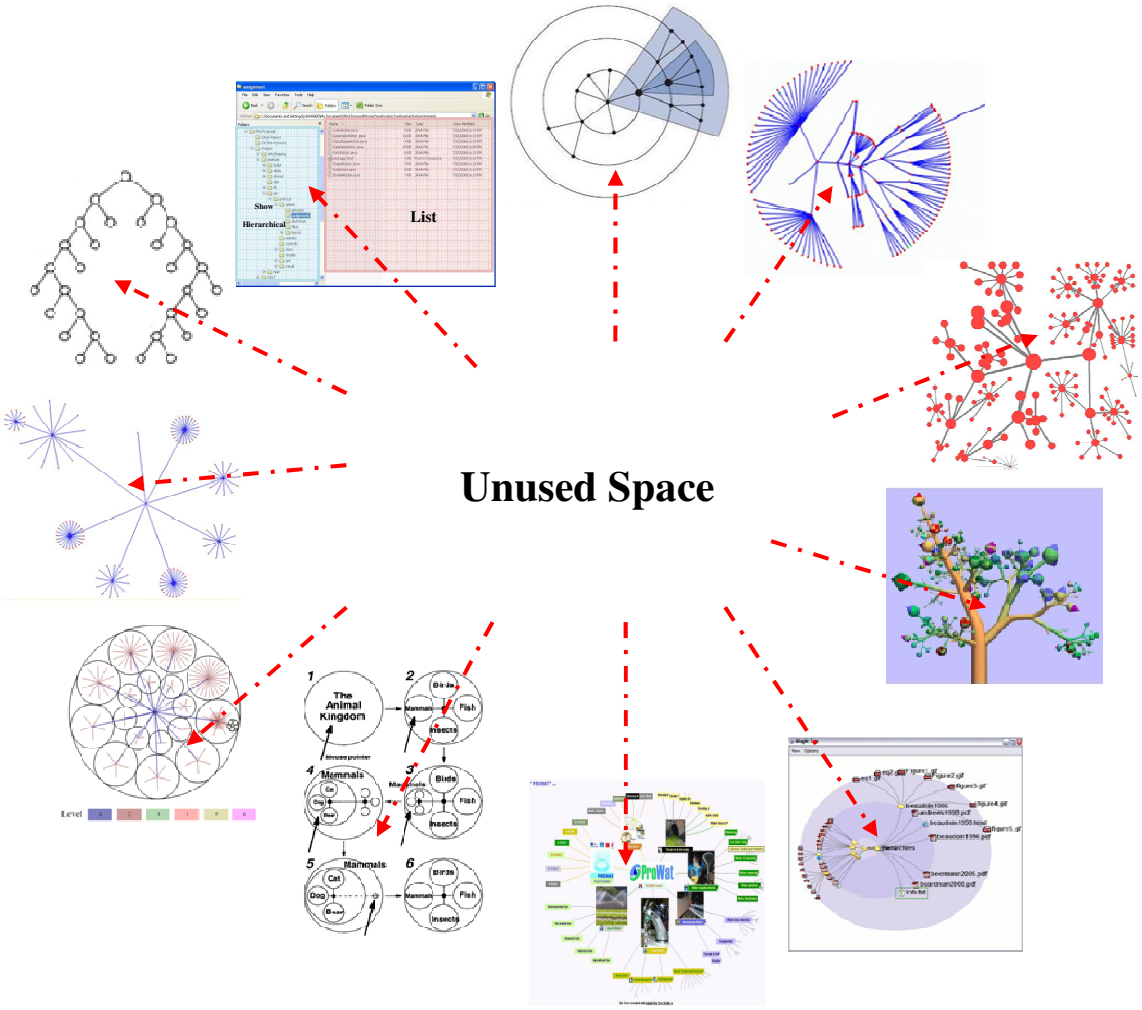


Figure 3.19. Illustration of Unused Space of Node-Link Diagrams

Therefore, none of these techniques can serve as a suitable temporary infovis technique. A new infovis technique needs to be invented from scratch. We skip the fourth step of evaluating temporary infovis and directly go to the step of new infovis technique invention.

3.3.4 Original Radial Edgeless Tree

This section introduces a hierarchy visualization technique, RELT [47] [48], which is especially designed for small screens.

3.3.4.1 Basic Idea

A rooted tree $T= (V, E, r)$ consists of a tree T and a distinguished vertex r of T as the root. Each vertex v has an associated value $w(v)$, which we call weight.

The entire rectangular display area is partitioned into a set of none-overlapping geometrical polygon nodes: $P(v_1), P(v_2), \ldots, P(v_n)$ that are used to visually represent vertices $v_1, v_2, \ldots, v_n$. Each polygonal node $P(v)$ is defined by three or four cutting edges which may be shared with other nodes. These boundaries are defined as below:

1. A common boundary shared with its parent represents the child-parent relationship.
2. A common boundary shared with all its children represents the parent-child relationship.
3. One or two boundaries shared with their siblings represent the sibling relationships.

The geometrical size of a node $P(v)$ is calculated based on its weight $w(v)$. Instead of using explicit edges or enclosure, RELT uses boundary-sharing to represent rational structure. Thus, the display space utilization is maximized. The entire tree T is drawn hierarchically from the northwest at the root to the southeast in the top-down manner, which naturally follows the traditional way of human perception of hierarchies. Note the approach can be easily adapted to move the root to other screen locations.

An intuitive method combining the previous two approaches is constructed with three steps. First, a normal connection-based method, the classical hierarchical view [103] for example, is applied. Second, each node is considered as a balloon and all the balloons are inflated until they occupy the whole screen. Third, these anomalistic non-overlapping balloons are relocated to simulate the tree structure. Although the above step appears like a space-filling approach, the result is more like a radial display. As presented next, the difference from the typical radial

approaches, such as InterRing [134], is that our approach computes area allocations based on the nodes' weights, rather than their angles.

3.3.4.2 Algorithm

For a given tree, the method recursively calculates the weight for each vertex. Vertices are classified into four types and each type is assigned a corresponding rule. The root is assumed to be located at the upper left corner. We employ the depth-first search to traverse the tree. Whenever a new vertex is met, the corresponding rule is applied. Every rule considers two operations: node area distribution operation and how to recursively divide its area for its children. After completely traversing the tree, the entire display area is partitioned into a set of non-overlapping polygons which are used to represent Vertices v_1, v_2... v_n. In this case, the display area is fully utilized and a set of graphical links that are commonly used in traditional connection-based methods are avoided. The algorithm is given in Peudocode 3.1as below:

```
procedure RELT (matrix ad_matrix)
begin
 Para_Creator (ad_matrix)
 // Calculate the necessary parameters for each node.
 DFS (ad_matrix)
 // Depth first search to traverse the tree
 if vertex v is new then
    int L= Test (v)
    // Return rule L to v
    Polygonal_Node (v, L)
    // Assign a region to v with rule L
    Partition_Area(v)
    // Divide the area depending on the weights of v's
        children
    fi
end.
```

Peudocode 3.1. Algorithm of Original Radial Edgeless Tree

The RELT algorithm shown above consists of four major functions.

Para_Creator (matrix ad_matrix) calculates the necessary parameters, including weight, depth, parent and children for each vertex. A vertex v is assigned with a weight $w(v)$, which is calculated in the following way:

- If vertex v is a leaf, $w(v) = 1$.
- Otherwise, if v is not a leaf and has m children $\{v_1, v_2, \ldots, v_n\}$ then

$$w_N = 1 + \sum_{i=1}^{m} w_i$$

Equation 3.1. Vertex Weight Calculation

Test (vertex v) returns the rule that should apply to vertex v. We classify all vertices into four types according to their characteristics in the tree. Specifically, a vertex v is of type:

1. If v is the only child of its parent.
2. If the parent of v has more than one child AND v is the first child of its parent.
3. If the parent of v has more than one child AND v is the last child of its parent (Note that child vertices are numbered from left to right. The left most child is the first and the right most child is the last).
4. If the parent of v must have more than one child AND v is neither the first nor the last child of its parent.

Before defining the function *Polygonal_Node (Vertex v, int L)*, several notations and definitions should be introduced.

- L_{Ni}: is the i_{th} cutting edge of node $P(v)$. The cutting edges are numbered from left to right, as illustrated as Figure 3.20.
- N_{NAS}: N's nearest ancestor with sibling(s) as illustrated in Figure 3.21 nearest ancestor of N that has at least one sibling.

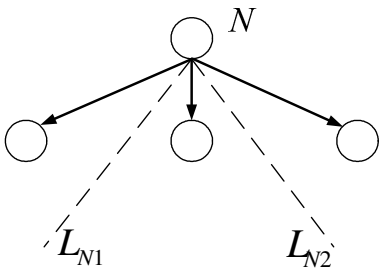


Figure 3.20. Cutting Edges

- N_{NAS_P}: The parent of N_{NAS}.
- N_{NASL}: N_{NAS} that is not the leftmost sibling.

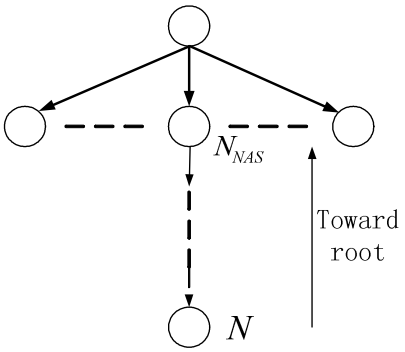


Figure 3.21. *N*'s Nearest Ancestor with Siblings

- N_{NASL_P}: The parent of N_{NASL}.
- N_{NASR}: N_{NAS} that is not the rightmost sibling.
- N_{NASR_P}: The parent of N_{NASR}.

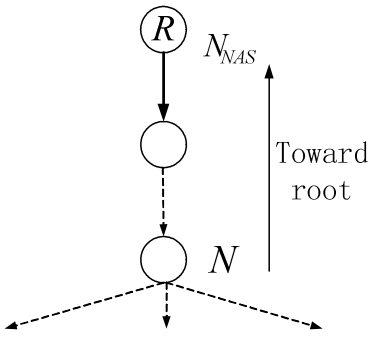


Figure 3.22. Special Case with the Root being Named N_{NAS}

Consider such as case, as illustrated in the Figure 3.22, where a node N has no N_{NAS}. We treat it as a special case by making the root R as N_{NAS}.

Polygonal_Node(Vertex v, int L) represents a vertex in a polygonal shape, rather than a rectangular or circular shape that is commonly used in other connection-based visualizations. Each node is constructed by linking two division lines, so this function essentially dictates how to choose these two cutting edges.

1. $L = 1$ (Node type 1)

   The cutting edges used by node $P(v)$ are the same as those by N_{NAS}. Rule1 first finds cutting edges used by N_{NAS} and then links them together to form *N's region*.

2. $L = 2$ (Node type 2)

   The first cutting edge chosen by Rule2 is $L_N{}^1{}_i$ where $N^1 = N_{NASR\_P}$ and N_{NAS_P} is the $(i+1)_{th}$ child of N_{NASR_P}. The second cutting edge is where N^2 is the parent of N.

3. $L = 3$ (Node type 3)

   The first cutting edge selected by Rule3 is $L_N{}^1{}_i$ where N^1 is the parent of N. The second cutting edge is $L_N{}^2{}_i$ where $N^2 = N_{NASL\_P}$ and N_{NASL} is the $(i+1)_{th}$ child of N_{NASL_P}.

4. $L = 4$ (Node type 4)

   The two cutting edges chosen by Rule4 are $L_N{}^1{}_{(i-1)}$ and $L_N{}^1{}_i$ where N^1 is the parent of N and N is the i_{th} child of N^1.

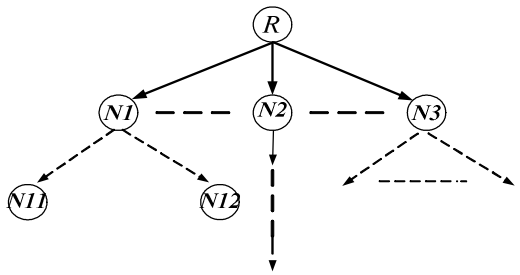


Figure 3.23. An Example Tree

Partition_Area (Vertex v). This function divides the remaining area of a vertex *v* based on the total weight of its children. The partitioned areas are allocated for the branches rooted at *v*'s children.

Figure 3.23 shows an example tree. *R*, the tree's root, partitions the whole 90 degree angle to its children P(*v1*), *P(v2)* and P(*v3*), the shade area in Figure 3.24 is the area given to the branch rooted at *N1*. *N1* uses cutting edge $L_{N1,1}$ to recursively partition the area into *N11* and *N12* surrounded by a dashed line. Because *N11* and *N12* have the same weight by Equation 3.1, they occupy the same-sized areas.

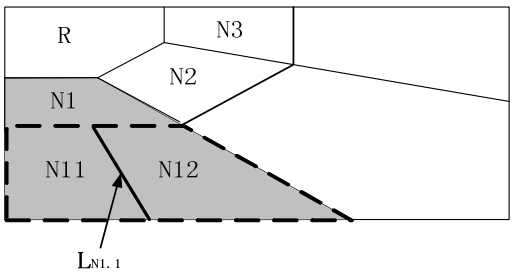


Figure 3.24. Drawing of the Branch for the Tree in Figure 3.23

3.3.4.3 Complexity Analysis

For an n-node tree, the complexity of function *Para_Creator* is O(n^2) because the adjacent matrix is used for information structure storage. Depth-first search is used to traverse the tree; for each new node, functions *Test(), Polygonal_Node()* and *Partition_Area()* are applied. All these three functions are O(*1*). So the complexity of this function is O(n^2).

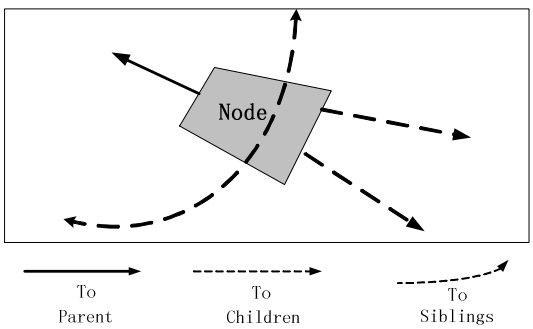


Figure 3.25. General Directions in Parent-Child Relationships

Using the above RELT algorithm, the tree structure is clearly displayed, as shown in Figure 3.25. For any node, its parent is in the upper left direction, and its children are in the lower right direction. The nodes at the same level are located along the dashed curve. In addition, this method recursively divides the whole display area, maximizing the screen usage. The next section presents an application of the algorithm.

3.3.4.4 A Case Study: Music Selection

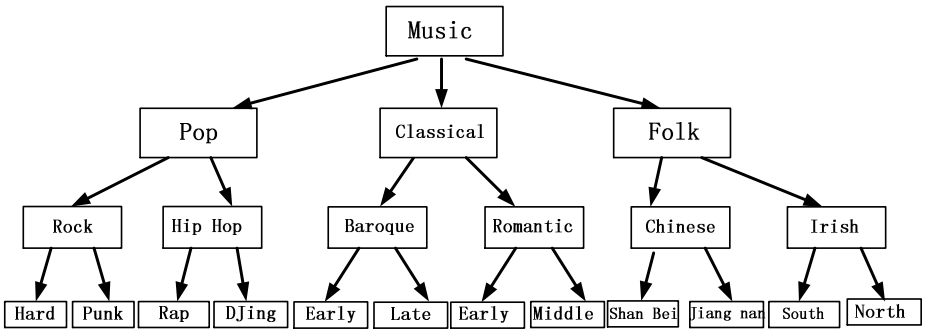


Figure 3.26. A Music Classification Example

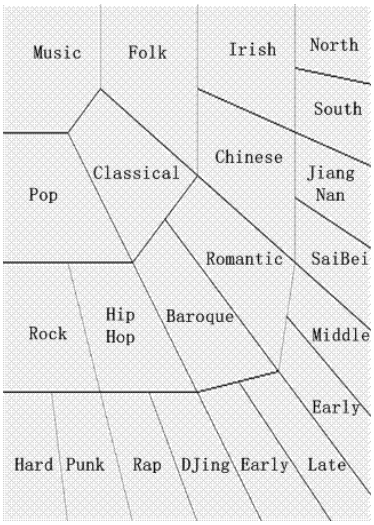


Figure 3.27. RELT for the Example Music Classification in Figure 3.26

RELT works well for hierarchical information, especially with the overall structure revealed on a limited screen estate. One of the current trends is to combine mobile phones with an MP3 player.

With the available storage capacity and improved sound effect, consumers can download many music pieces from the Internet. With the increasing number of music selections available on the Internet, there is an urgent need for a commonly accepted music classification system that can assist navigation and selection. The most common approach involves using a menu bar. The structure of a menu bar is very simple, like artist – album – track, and may be defined by users (IPod, for example). It however does not show clearly the structure of the music categories. Figure 3.26 shows an example music classification, whose RELT display is shown in Figure 3.27.

This method works well when the input tree is almost balanced. The smaller the difference between $w^{max}(l)$ and $w^{min}(l)$, the maximal and minimal weights at level l, the more balanced and esthetic is the tree.

The example shown in Figure 3.26 is totally balanced because all the nodes on the same level have the same weight. Figure 3.28 gives an unbalanced tree since the node "Folk's" (in a shaded ellipse) weight is 3 which is smaller than 5 of node "Classical" (in a white ellipse). This leads to an unesthetic layout, as shown in Figure 3.29. Nodes "Irish" and "Chinese", in shaded ellipses, are long, across several levels. Those long nodes make the hierarchical levels unclear. The next section discusses an optimization technique that reduces the number of such long nodes.

3.3.4.5 An Optimization

In an unbalanced tree, some leaf nodes may over-represent their areas. More specifically, such nodes take more than one level in the final RELT representation.

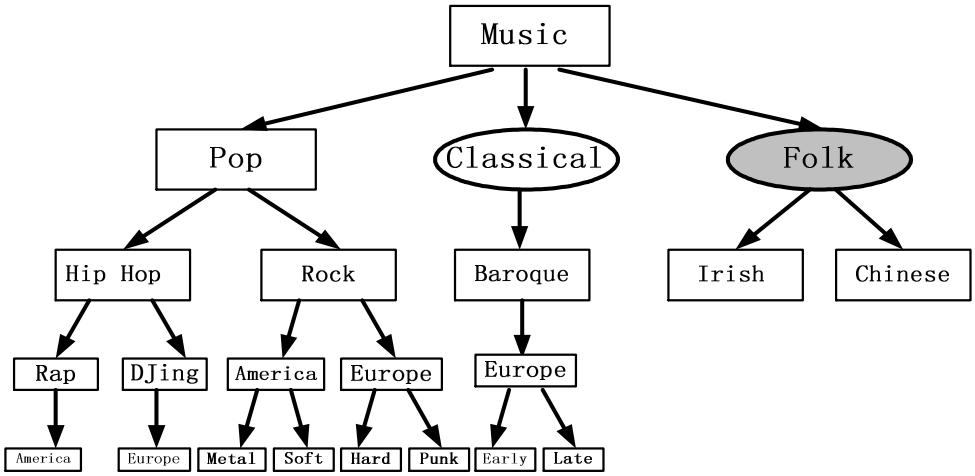


Figure 3.28. An Unbalanced Tree

Take the node "Chinese" in Figure 3.28, for example; it is a level 3 node but it covers across levels 3 to 5 as shown in Figure 3.29. The following subsections first introduce the notations and a layout estimate function, and then present the optimized algorithm in details.

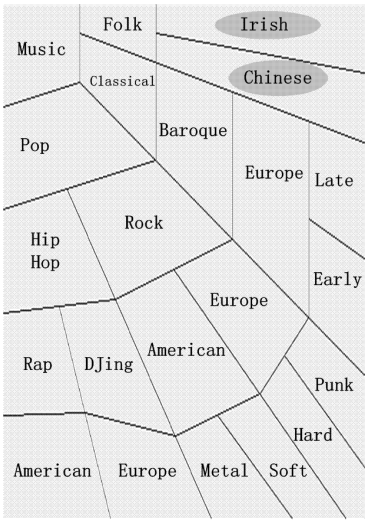


Figure 3.29. RELT for the Example in Figure 3.28

3.3.4.5.1 Estimating Node Overrepresentation

The greater the number of levels over-represented by nodes in a RELT layout, the greater the misunderstanding of the tree structure the layout may lead to. The total number of over-

represented levels is therefore a critical criterion in estimating the effectiveness of RELT for easy human perception.

Definition 1: Assume each leaf node is counted once; the *depth* of a node *N*, denoted *N.depth*, is the number of nodes from *N* (including *N*) to its nearest leaf.

Definition 2: A leaf node *N* is over-represented iff *N.depth* < *i.depth*, where *i* is another leaf node. We denote the set of over-represented leaf nodes as *OR*.

According to the RELT algorithm, non-leaf nodes will never over-represent. Every node in OR shares at least one of its boundaries with some other nodes at more than one level. For example, node "Chinese" in Figure 3.28 is in OR because it shares one of its boundaries with "Classical," "Baroque" and "Late" that are at different levels. Node "Irish" is not in *OR* because it shares each boundary with exactly one other node.

Definition 3: The number of *levels* over-represented by node *N* (in *OR*) is denoted L_OR_N. If both side boundaries of *N* are shared with *N*'s sibling nodes, L_OK_N is the sum of the levels on these two boundaries.

The estimation-function NOR for overall node over-representation can be constructed as follows:

$$NOR = \sum L\_OR_N$$

Equation 3.2. Overall Node Over-Representation Calculation

Clearly, a small *NOR* is desirable.

3.3.4.5.2 Minimizing Overrepresentation

We consider the state of a tree as the one that uniquely determines the parent-child relationships and ordered (left to right) sibling relationships. For example, exchanging a node's left and right children will change the tree state. A change of relative positions of any two nodes will change the tree's state. A given state of a tree uniquely determines its layout by the RELT algorithm.

To obtain the best RELT layout, we need to investigate how to obtain the best state of a tree. A tree's nodes are initially divided into groups according to their levels as described in Section 3.2. At each level, the nodes are numbered from left to right. In the following, we will use *TN_Depth(li)* to represent a function that measures the maximum difference between the depths of any two leaf nodes at a given level *i* for a tree state.

Let

- N_{ij} be the node at level *i* numbered *j*.
- $N_{ij}.depth$ be the depth of node N_{ij}.
- l_{num} be the number of levels of *T*.
- l_{i_num} be the number of nodes at level *i*.

$$\left|N_{i(i+1)}.depth - N_{ij}.depth\right| = \begin{cases} \left|N_{i(i+1)}.depth - N_{ij}.depth\right| & (1) \\ 0 & (2) \end{cases}$$

$$TN\_Depth(l_i) = \sum_{j=1}^{l_{i\_num}-1} \left|N_{i(i+1).}.depth - N_{ij}.depth\right|$$

Equation 3.3. Calculation of *TN_Depth(li)*

(1). If one of N_{ij} and $N_{i(j+1)}$ is a leaf and the other is a non-lead node.

(2). If both N_{ij} and $N_{i(j+1)}$ are leaves or are non-leaf nodes.

The following function measures the maximum difference in depth between any two leaf nodes for a tree state:

$$TN\_Depth(State) = \sum_{j=1}^{l_{i\_num}-1} TN\_Depth(l_i)$$

Equation 3.4. Calculation of TN_Depth(State)

Theorem 1: Let s_{old} and s_{new} be the old and new final states, and L_{old} and L_{new} be the corresponding old and new layouts of a tree; if $TN\_Depth(s_{new})$ is smaller than $TN\_Depth(s_{old})$, then $NOR(L_{new})$ is also smaller than $NOR(L_{old})$.

Proof: According to Equation 3.4, the depth difference between two adjacent nodes contributes to the function only if one node is a leaf and the other is a non-leaf node. Thus the leaf node is over-represented in the layout and the depth difference is exactly the number of levels over represented. For a given state of a tree and its layout, the values of *TN_Depth* and NOR are the same. The only difference is that *TN_Depth* explores estimates based on the tree's state and NOR estimates the outcome layout.

Now the strategy of minimizing overrepresentation becomes how to use function *TN_Depth(State)* to find the state that can result in the best layout of a tree. For a tree, the best state s_{BEST} satisfies Equation 3.5:

$$TN_{Depth}(S_{BEST}) = \min_{s_i \in S} TN\_Depth(S_i)$$

Equation 3.5. Description of the Best State

where *S* is the set of all the tree's states.

Next, finding the minimum number of *TN_Depth(State)* becomes the key. This can be done in two steps. The first step is expressed in Equation 3.3 and the second step in Equation 3.4. To simplify the problem, a basic assumption is initially constructed. Equation 3.4 will achieve its minimum point when all the results for each level in Equation 3.3 are the smallest. The process is then simplified to how to obtain the smallest number for each level in Equation 3.3 as described below.

Definition 4: A unit is the set of all the siblings who have the same parent.

A unit's position at its level is the number counted from left to right. Let U_{ij}, denote the unit at level i and position j. For example, in Figure 3.29, "Hip Hop" and "Rock" make up a unit U_{31}, "Classical" is a unit U_{32} by itself, and "Chinese" and "Irish" make up the unit U_{33}.

Assume level i contains m units and each unit has exactly two nodes. A naive way to obtain the best layout is to compute overrepresentation by Equation 3.4 for the all the unit combinations and choose the state with the smallest depth. The complex is $O(2^m)$ because there are two possibilities in each unit. It is explicitly not a good choice especially when m is large.

Theorem 2: For a single unit, Equation <3> derives the minimum number if the nodes are sorted by their depths.

Proof: If there is initially a sorted unit including m nodes, then two nodes D_i and D_j are exchanged.

U_{SORT}:

$$(D_1,D_2\ldots D_{i-1}, D_i, D_{i+1}\ldots D_{j-1}, D_j, D_{j+1}\ldots D_m)$$

$$1 \le h \le t \le m \quad D_h \le D_t$$

U_{UNSORT}:

$$(D_1,D_2\ldots D_{i-1}, D_j, D_{i+1}\ldots D_{j-1}, D_i, D_{j+1}\ldots D_m)$$

$$TN\_Depth(U_{SORT}) - TN\_Depth(U_{UNSORT})$$

$$=(| D_i - D_{i-1}| + |D_{i+1} - D_i| + |D_j - D_{j-1}| + |D_{j+1} - D_j|) -$$

$$(|D_j - D_{i-1}| + |D_{i+1} - D_j| + |D_i - D_{j-1}| + |D_{j+1} - D_i|)$$

$$= (D_{i+1} - D_{i-1} + D_{j+1} - D_{j-1}) - (2D_j - D_{i-1} - D_{i+1} + D_{j-1} + D_{j+1} - 2D_i)$$

$$= 2(D_i + D_{i+1}) - 2(D_{j-1} + D_j) \ge 0$$

$TN\_Depth(U_{SORT}) = TN\_Depth(U_{UNSORT})$ *becomes true when* $i + 1 = j$ *or* $D_i = D_j$.

Now we can derive an optimized RELT algorithm. Based on Theorem 2, function Sort (Node *N*) is inserted between Polygonal_Node(Node *N*, int *L*) and Partition_Area(Node *N*).

For every new non-leaf node *N,* function Sort (Node *N*) sorts its children by their depths. Assuming the j_{th} non-leaf node has n_i children, the complexity of function Sort (Node *N*) is $\sum_{j=1}^{n_{in}} n_j \lg n_j$

Proof of complexity is given as follows:

$$\sum_{j=1}^{n_{in}} n_j \lg n_j \leqslant \sum_{j=1}^{n_{in}} n_j \lg n_{in} \leqslant \lg n_{in} \sum_{j=1}^{n_{in}} n_j < n \lg n_{in} < n \lg n$$

So, the complexity of the optimized algorithm is O(n^2) +O(*nlogn*).

3.3.5 Generalized Radial Edgeless Tree

3.3.5.1 Methodology and Terminology

Figure 3.30 illustrates a university's web site, where schools of "Management," "Engineering," and "Science" each have a few departments. The original RELT [47][48] algorithm fixes the location of the root ("University") at the top-left corner of the screen as shown in Figure 3.31 (a).

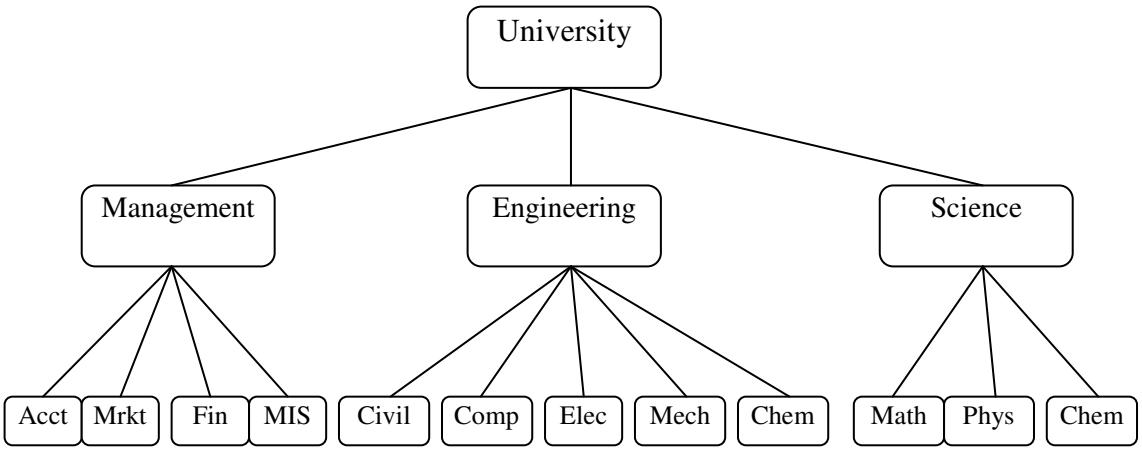
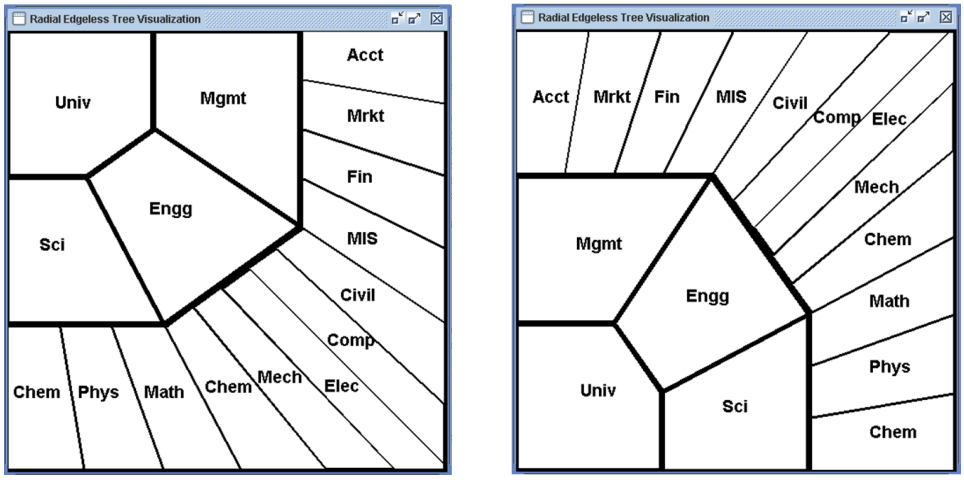


Figure 3.30. A University Web Structure

This fixed layout certainly cannot fit every different application. General RELT [49][50] allows the root to be displayed anywhere on the screen, depending on the user's preference and the requirements of different applications. For example, with the generalized version, it should be straightforward to display a layout with the root at the bottom-left corner (Figure 3.31 (b)) or one of the other corners.

Figure 3.31 (a) and (b) are in center-rooted layout, where each level-2 node, i.e., a school, shares a border with the root, and each level-3 node shares a border with its level-2 parent. We will call this type of center-rooted layout as concentric, or simply CC. As shown in Figure 3.31 (a) with the university example, departments are located under the school they belong to, which in turn locates adjacent to the university.

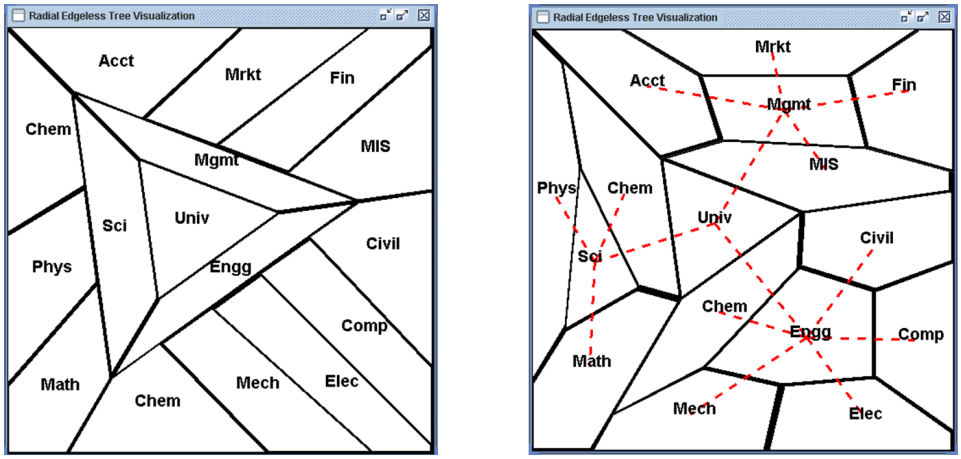


(a) Original RELT layout with root at top-left

(b) An alternative layout with root at bottom-left

Figure 3.31. University Structure Visualized as a Rooted Tree in RELT

Figure 3.32 (b) illustrates another type of center-rooted layout which considers the sub-area of each child node as a center-rooted sub-tree. We will call this latter type of layout as multi-centric, or simply MC. In this case, the schools form the sub-centers with their departments around them.



(a) Concentric layout (b) Multi-centric layout

Figure 3.32. Variations of Center-rooted Visualization

Given any of these layouts, the user is able to navigate through the hierarchy structures. For example, to view the detailed information on the Department of Computer Engineering, the user can select "Comp". By selecting "Comp" in Figure 3.31 (a) that has the root at the top-left, we obtain the zoom-in view as in Figure 3.33, where the entire display space is filled by the sub-tree rooted at "Comp". Therefore with a single user interaction, multiple levels of hierarchy (five levels in the university example) can be reached.

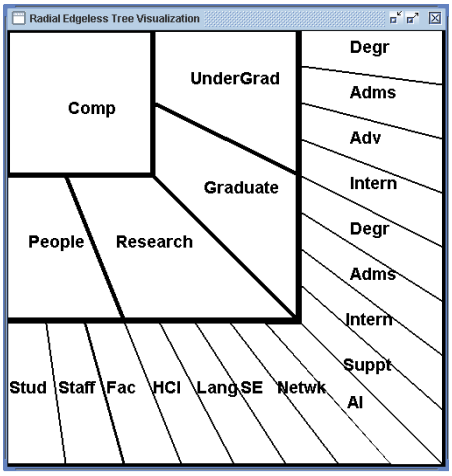


Figure 3.33. A Navigated View of Two Levels Down the Hierarchy

How a node is selected depends on the interaction technique that is supported for navigation. A touch-screen interface (by either a finger or a stylus) would apparently suit the RELT

methodology very well. A single touch on a node would serve the selection of the node. With a key or button-based interaction support, direction buttons may be assigned to the moving directions within a predefined radial range, and/or up and down the hierarchical levels, plus a confirmation button, as we experimented on a cell phone emulator to be discussed in Section 5.

In general, the number of levels on a single display can be determined based on the application, or maximized to the extent that the polygonal nodes have enough space for text labels. If color coding can easily discriminate and identify multiple groupings and levels of information, each tree node may be drawn by just a few colored pixels and thus maximize the size of the hierarchy being visualized. Therefore, RELT methodology has great scalability.

Let us now discuss the concept and terminology of generating the general RELT layout. We will refer to a tree node as a vertex that occupies a polygonal area in the drawing space.

General RELT classifies vertices into three types:

1. $v = r$: $T(v)$ represents the entire tree.
2. $v \neq r$ AND $v \notin L$: $T(v)$ represents the sub-tree rooted at v.
3. $v \in L$: $T(v)$ refers to the vertices r.

If $v \notin L$, i.e., type 1 or 2, then the set of v's children can be expressed by: $CV = \{u \mid (v, u) \in E\}$. CV_i is the i_{th} child of $_v$. The children are sorted according to the order of their appearance in the input data.

The display space is recursively divided into several non-overlapping polygonal areas each visually representing a vertex. Three types of "area" are defined for the general RELT:

1. $WA(v)$ represents the entire area under vertex v.
2. $OA(v)$ is the area occupied by v itself.

$DA(v)$ is the area occupied by all of v's descendants.

3.3.5.2 Rules and Algorithm

A RELT layout can be generated by applying the following four general rules:

1. Every *T(v)* has *WA(v)* where *T(v)* should be drawn. *WA(v)* is assigned to *v* by its parent. *WA(r)* is the entire display area.
2. How *OA(v)* is determined inside *WA(v)* depends on the layout type *(LT)* specified by the user.
3. If *v* is a non-leaf vertex, it distributes its area *DA(v)* to its children.
4. The area size of a leaf vertex is proportional to one of the vertex's properties, such as the weight.

```
Algorithm RELT
Input: vertex set V, root r, layout type LT
Begin
Set root location // User clicks on screen to
                     decide root location
DFS       // Depth first search to traverse the tree
if vertex v is not yet processed then
   if vertex is a non-leaf
      OA(v) = createOA(v, LT);
      // calculate the area occupied by v
      if v has more than one child
         DA(v) = WA(v) - OA(v)
         WA(CV) = DF(DA(v))
          // v distributes DA(v) to its children
      else    // v has only one child
         DA(v) = WA(v)
          // DA(v) belongs to the only child
   else    // leaf vertex
         WA(v) = OA(v) = DA(v)
```

Peudocode 3.2 Algorithm of General Radial Edgeless Tree

A RELT layout can be calculated by function *RELT(V, r, LT)* in which *V* is the vertex set, r specifies the root vertex, and *LT* indicates the layout type. The user-provided *LT* parameter determines whether the hierarchy is viewed concentrically, i.e., the CC layout, or children

surrounding their parents that form multiple sub-centers, i.e., the MC layout, as introduced in Section 3.1. The RELT algorithm is presented in pseudo code above.

Each vertex v is assigned a corresponding weight $w(v)$ which is computed by a weight function WF:

$$w(v) = WF(w(CV_1), w(CV_2)\ldots\ w(CV_n))$$

Equation 3.6. Weight Calculation in General Radial Edgeless Tree

Equation 3.6 basically says that the value of $w(v)$ depends on the overall weight of v's children. Different weight functions may be defined to meet different application requirements (see Section 4.2 for the weight function used to visualize the stock market). This dissertation uses the following simple weight function as an example:

1. If $v \in L$, $w(v) = WF(1) = 1$;
2. Otherwise, assuming the given v has n children, then

$$w(v) = WF(1, w(CV_1), w(CV_2)\ldots w(CV_n))$$

$$= 1 + \sum_{i=1}^{n} w(CV_i)$$

A leaf vertex v has no children, therefore requires no distribution. Only its own area $OA(v)$ needs to be calculated. For a non-leaf vertex v, after $OA(v)$ has been constructed, $DA(v)$, is distributed to its children. The distribution is determined by a distribution function $DF()$. Here, we propose $DF()$ that partitions the distribution area based on the size of $DA(v)$ and the children's weights:

$$DF(DA(v), w(CV_1), w(CV_2)\ldots w(CV_n)) \rightarrow WA\ (CV)$$

Equation 3.7 Distribution Function in General RELT

Function $createOA(v, LT)$ creates the node v's own area, depending on the layout type LT. The following pseudo code describes $createOA(v, LT)$ for the concentric layout type. Due to the space limit, the function description for the multi-centric layout is omitted here.

```
Function createOA(v,LT)
NLevel = the number of levels in the tree
If v = r has m children (m >1)
   // rootP is the root location determined by the user
   // startP is a point on the boundary of the screen
   // In our implementation, we fix startP = (0,0)
   movingP = startP
   draw a line scanL from rootP to movingP
   no = 0
   record scanL as Lno
   while (no < m)
      movingP along boundary clockwise, accumulate area by
  scanL
          If accumulated area is to the right child depending
    on WF()
              no = no +1
              record scanL as Lno
  while (no >0)
      calculate Pno
      // distance from rootP to Pno is 1/NLevel of Lno
      no = no -1
   while (no < m)
      link Pno and P(no.-1 mod m) to form OA (r)
      no = no +1
If v = r has one child
   draw OA (v, LT) as a polygon
If v is non-root and non-leaf
    // vParent is parent of v, vLevel is level of v
    // commonL is the line shared by WA(v) and OA(vParent)
    // point1 and point2 are two end points of commonL
    point1' = point1
    point2' = point2
    // boundary1 is boundary of WA(v) containing point1.
    // boundary2 is boundary of WA(v) containing point2.
    move point1' and point2' along boundary1 and boundary2
    if distance from point1' to point1 is 1/vLevel of the
length of   boundary1, then stop (similarly for point2')
    connect point1' and point2' to form stopL
    // OA(v) is the area between commonL and stopL.
end
```

Peudocode 3.3. Algorithm of Function *createOA(v,LT)*

We now discuss the complexity of the RELT algorithm. As per previous discussion, a given tree $T = (V, E, r)$ has n vertices which are divided into three types. RELT applies depth-first search to traverse the tree.

Table 3.2. Vertex Types and Corresponding Operations

Vertex Type	Operation
1: $v=r$	*createOA*(v, *LT*) DA(v) = WA(v) – OA(v) DF (DA(v)) -> WA(CV)
2: $v \neq r$ AND $v \notin L$	createOA(v, *LT*) DA(v) = WA(v)
3: $v \in L$	WA(v) = OA(v) = DA(v)

Table 3.2 illustrates the vertex types and their corresponding operations used in the RELT algorithm. All the operations in the right column can be computed in linear time. Therefore the overall time complexity is O (n), where n is the number of vertices.

In summary, RELT recursively partitions the display area into a set of non-overlapping polygons. Because every part of the screen is used, an economic screen estate is achieved. Parent-child relationships are explicitly represented by adjacent relationships, and thus the structural clarity is maintained.

3.3.6 Evaluation of RELT

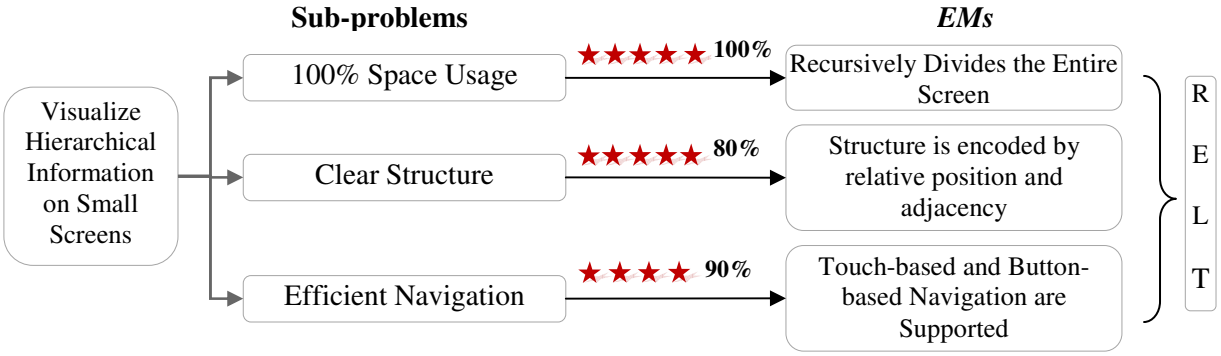


Figure 3.34. Evaluation of RELT

Figure 3.34 illustrates how RELT solves the TP: Visualize Hierarchical Information on Small Screens. The percent number behind each sub-problem denotes *dos* for its associated *EM*. RELT recursively divides a screen into a set of non-overlapped polygons. Each polygon denotes a node in the represented hierarchy so that the screen is 100 percent utilized. Unlike Treemap which uses enclosure to represent parent-child relationships, RELT uses relative position and adjacency of polygons to represent hierarchical relationships. RELT generates a very esthetically pleasing layout for a small-sized balanced tree. However, the beauty of layout decreases when RELT is applied on a huge hierarchy, especially on ones with unbalanced structure; therefore, *dos* is deducted by 20 percent. Touch-based and button-based navigation are both supported by RELT. Ten percent is cut for *dos* due to the lack of smooth transition when navigating a hierarchy. The *DoS* of RELT for *TP* is calculated as follows:

$$Dos = \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k}$$

$$= \frac{5*1+5*8+4*0.9}{5+5+4}$$

$$= 90\%$$

Equation 3.8. *DoS* of RELT

3.4 Case Study One: RELT Stock Visualization

3.4.1 Current Approaches

Treemap is a space-constrained and rectangular space-filling technique for visualizing hierarchical information. Its desirable features, such as economic screen usage, attract a wide range of commercial usage.

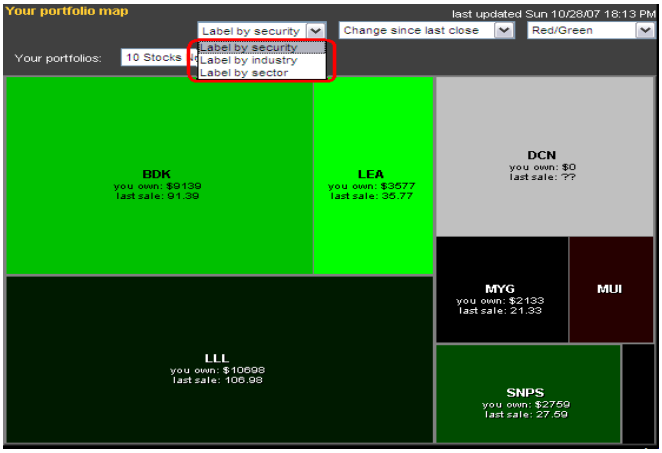


Figure 3.35. Treemap for Stock Market Visualization

Portfolio map [150] in Figure 3.35 visualizes "10 stocks now 799". Traditional stock coloring indicates stock price performance. Green stands for price up and red for down, black indicates an unchanged price, and gray means that users do not own any shares. The rectangle size indicates the market capitalization of the corresponding company.

"I Deal Solution" [143], shown in Figure 3.36, is a visualization tool applied on the data from the New York Stock Exchange website. The intrinsic metaphor used includes sphere size corresponding to the stock price, color to the change of price, vibration to the percentage of change, and sphere location to the transaction volume.

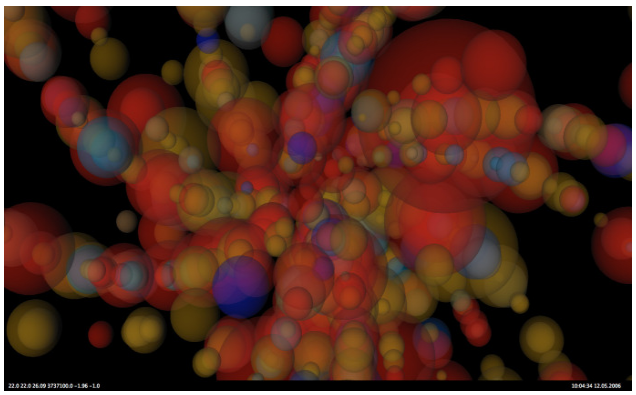


Figure 3.36. I Deal Solution - 3D Sphere-based Stock Visualization [143]

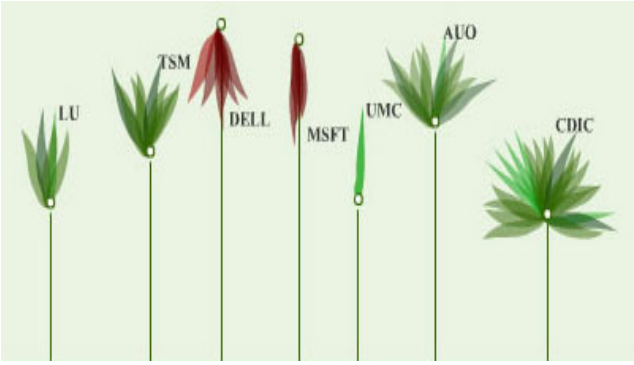


Figure 3.37. Stock Market Ticker Garden

Figure 3.37 illustrates the Stock Market Ticker Garden [152]. This visualization provides an easy understanding for monitoring stock portfolios. Stocks' real-time performance is visually expressed by the flower's color, height, and radian of blossom. The price is illustrated by the flower's height. How much does the flower's blossom indicate the percentage of price change? The flower's color and the blossom direction express the ascent or descent of the stock's price compared to the price of the previous trading day.

Stock Market Planetarium [153] in Figure 3.38 visualizes the real-time stock market activity as the night sky with stars glowing as trading happens on specific stocks. Slowly drifting stars represent the trading companies and gravitationally attracts other stars clot with similar stock price histories.

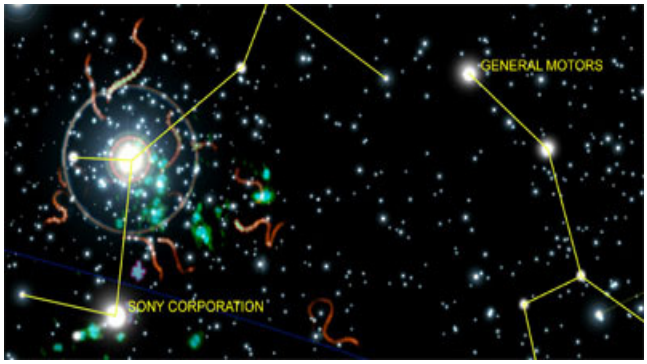


Figure 3.38. Stock Market Planetarium

These above visualization methods all have their individual appealing characteristics. They are, however, not suitable for stock visualization on small screens due to the following limitations:

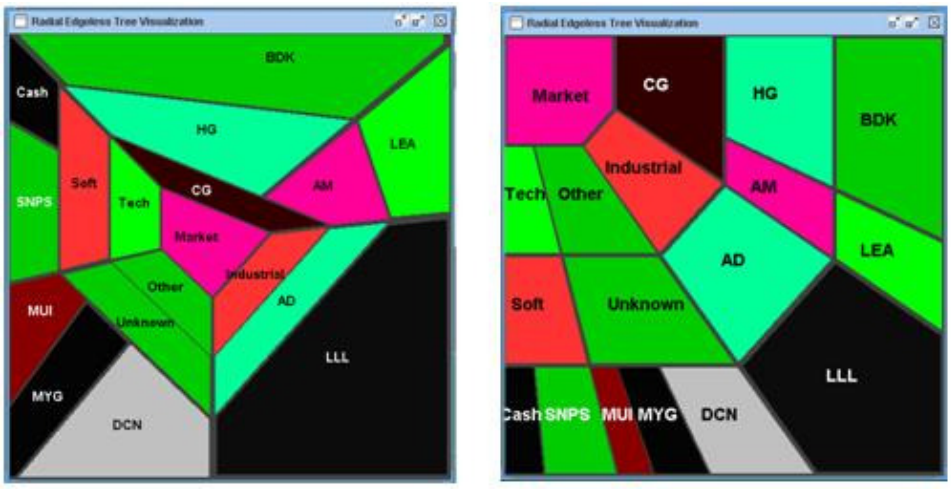
- Algorithms, such as 3D Sphere-based methods and Stock Market Planetarium, are designed for visualizing a huge data set. Personal mobile devices are not usually used to keep track of massive amount of information. Users are most likely disoriented with an overwhelming amount of information, especially on a small screen.
- The use of visual metaphors, such as 3D spheres, flowers, and stars, requires an intelligent and intuitive mapping of graphical attributes to the stock market parameters. An initial learning curve for new users is unavoidable.
- None of the approaches except the Treemap emphasizes the economic usage of the screen estate.
- These methods visualize each stock or security independently and locate stocks and securities based on their individual properties. None of them offers a clear classification structure for the market under monitoring.
- "How does the market look like today" is the essential question that a stock visualization tool needs to answer. What makes this question really difficult is that a good answer must contain both macro and micro views. Neither simply answering the market as a whole is up, nor answering a certain stock is down will satisfy the user. A good answer may be like: the whole market rallied but technology stocks were down, especially the software industry stock; "SNPS" (Figure 3.35), however, performs well among the software stocks. None of the aforementioned approaches can offer simultaneous display of both a big picture and individual stock details.

3.4.2 The RELT Solution

RELT offers some alternative solutions that overcome the major drawbacks of these approaches. By recursively partitioning the display area, every part of the screen is assigned to a vertex, so more economic space utilization is achieved. The industry-accepted convention of the stock coloring scheme can be easily applied in RELT to indicate the stock performance, in the same fashion as the one in the Treemap in Figure 3.35. As a case study, a typical stock market classification structure is illustrated in Figure 3.11.

We would like to visualize the price change of each individual stock as well as the capitalization of the corresponding company. We can use the color scheme similar to the one used for the Treemap in Figure 3.35, and make the size of the polygonal vertex for each stock a function of the company's capitalization. The corresponding weight function can then be customized as follows: If $v \in L$, $w(v) = WF(Cpt(v)) = Cpt(v)$, where $Cpt(v)$ represents the capitalization of the company represented by v; Otherwise, assuming the given v has n children, then $w(v) = WF(1000, w(CV1), w(CV2)\ldots w(CVn)) = 1000 + \sum_{i=0}^{n} w(CV_i)$.

It is generally more difficult to tell the difference in the sizes between two arbitrarily shaped polygons than between two regular shapes such as rectangles. We therefore use the constant 1000 in Equation 3.6, instead of 1 as in Section 3.3.5.2, to emphasize the size difference, i.e., the difference between the companies' capitalizations. Giving more weight to a non-leaf vertex v would contrast the relative proportion of $DA(v)/WA(v)$. The results are visualized in Figure 3.39, which includes a layout view with the root at the center (a) and another layout view with the root at the top-left corner.



(a) The center-rooted concentric layout (b) The root at the top-left

Figure 3.39. A RELT Visualization of Stock Market

Figure 3.39 illustrates RELT layouts that maintain a clear structural view by arranging vertices in a radial manner. More importantly, RELT provides both a macro view, i.e., the overall performance of the stock market, sectors, and industries at a glance, and a micro view, i.e., individual stocks that can be easily zoomed in if needed.

We believe that RELT outperforms Treemap in visualizing the tree structure, since RELT shows multiple levels on a single display while Treemap shows only one level. This also implies that only one level of hierarchy may be reached (i.e., zoom it or out) by a user interaction such as a mouse click or button push. Multiple user interactions are needed to navigate from the high-level market overview to individual stock (company) performances.

3.5 Case Study Two: RELT on Current Cell Phone Interface

To evaluate whether the RELT methodology provides a more efficient interface for navigation, this section presents an experimental comparison of our implementation on an emulator with one of the currently used cell phone interfaces, Sprint PCS Vision Phone®. The comparison is performed for the RELT center-rooted layout since the Sprint top-level interface looks similar

and provides the navigation structure from the center as pictured in Figure 3.40. The hierarchical information to be navigated by both interfaces is the Sprint cell phone's 5-level functions and services as described below.

3.5.1 Sprint Interface

The Sprint interface has nine top level service categories, as shown in Figure 3.40(a). There are totally 43 second-level sub-categories under the nine services, and each second level sub-category may include 0 to 6 third level items or sub-sub-categories. A third level sub-sub-category may include a few items at the fourth level. The last level is the fifth level. The entire hierarchical structure can be represented as a tree that has about 150-250 nodes. Some of the nodes at levels 2 through 5 are leaves, which should be the cell phone functions that the user wishes to arrive at in the shortest time, or with the fewest user interactions.

The primary navigational interactions on the Sprint interface are made through the four directional buttons surrounding a central button "Menu OK" (shown in Figure 3.40(a)) that is for confirming a selection.

3.5.2 Emulated RELT

The RELT methodology has been implemented on a SUN cell phone emulator as shown in Figure 3.40 (b) that reads in the Sprint hierarchy structure as an XML file. The default selection is at the root, i.e., "Menu", at the beginning, and the selected node is highlighted in the inverse color as shown in Figure 3.40 (b). Each screen display shows three levels of hierarchy in this experiment, with the top-level view shown in Figure 3.41 (to save space, this and images in Figure 3.42 only show the display area rather than the whole phone set). A selection on a lower level node will make that node a root, whose child and grandchild nodes are then displayed as

illustrated in the two examples in Figure 3.42. The left image (a) shows when the level 2 node "Messaging" becomes the root, and the right one (b) shows the level 3 node "Sound" becoming the root.

(a) Sprint PCS Vision Phone® User Interface

(b) RELT Emulator

Figure 3.40. User Interface Comparison

The user moves the highlighted selection around the screen by pushing the four direction buttons. Placing four direction arrows centered at the currently selected node as depicted in Figure 3.41, pushing a direction button (e.g. top) will move the selection to the neighboring node that the (top) direction arrow passes through ("Tools" in Figure 3.41). This simple navigation rule applies to all the displayed nodes. Pushing the center button (half-colored in Figure 3.40 (b)) confirms the selection of the highlighted node, in a similar fashion as on the Sprint Interface.

The text labels are arranged to maximize space usage while avoiding cluttering, particularly the overlap with the polygon edges. We use a simple approach in labeling the almost arbitrarily oriented polygons. The approach finds the center point as well as the longest side of the polygon to be labeled. It then rotates the text label around the text's midpoint to the same angle as the longest side of the polygon. The label is then placed in the polygon such that its midpoint coincides with the polygon's midpoint. Though this approach may not generate perfectly esthetic labeling, it is simple and fast with quite satisfactory display results as shown.

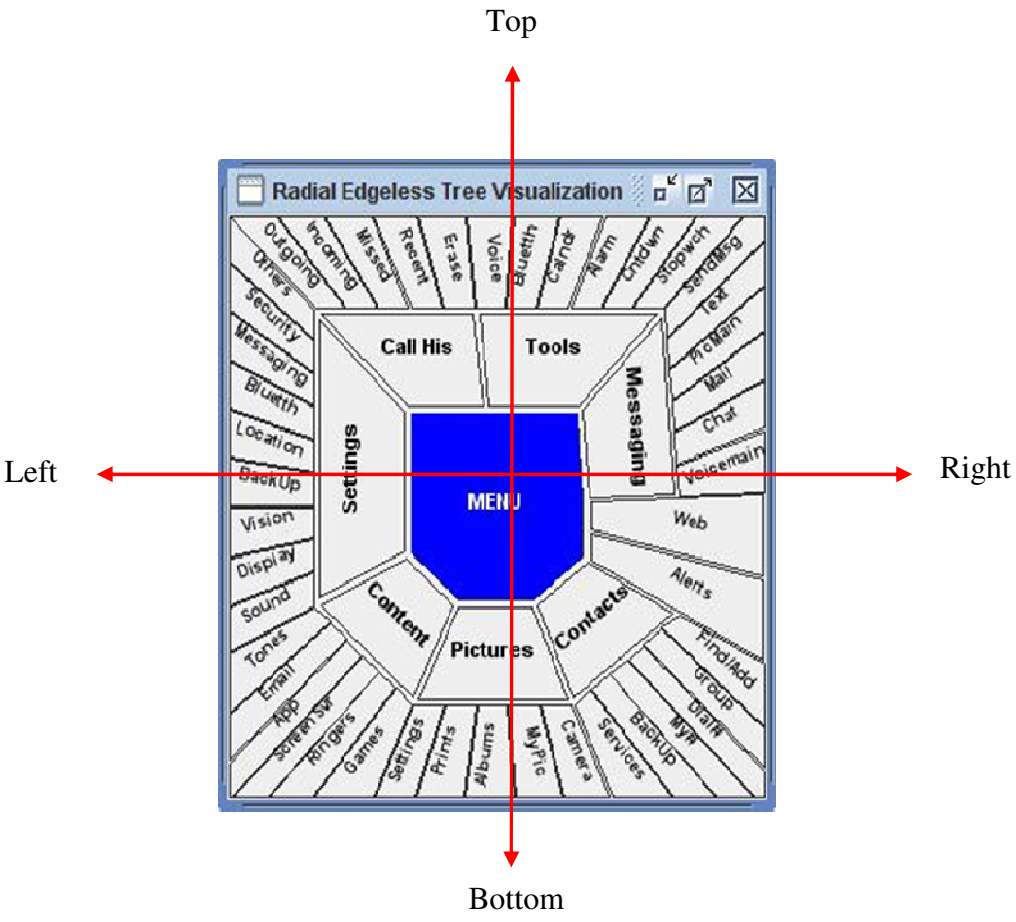
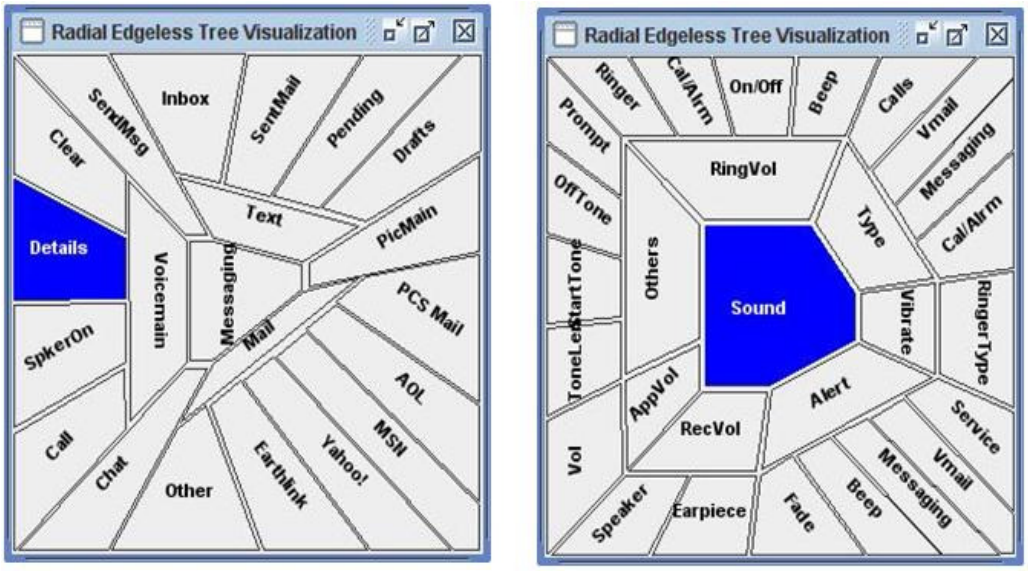


Figure 3.41. Top Level View of RELT with Assignments of Navigation Buttons



(a) RELT of levels 2, 3 and 4 (b) RELT of levels 3, 4 and 5

Figure 3.42. RELT on Different Levels

3.5.3 Result Comparison

This section compares the effectiveness of navigation for the Sprint and RELT interfaces. We first analyze them in an ideal case and then conduct an empirical study to examine the actual performance.

3.5.3.1 Analytical Comparison

Here we explain the inherent difference between these two methods. To faciliate the comparison, we model the problem with preconditions and assumptions.

- The example hierarchy can be represented as a full *k*-ary tree by applying traditional hierarchical drawing method, as illustrated in Figure 3.43. Assume a hierarchy of n levels.
- The number of levels that can be visualized effectively on one screen using RELT is limited at 3 ($\leq$ n). In reality, users are allowed to adjust the number of levels shown on the screen.

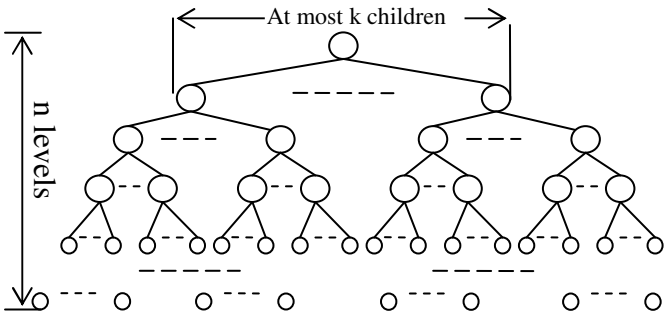


Figure 3.43. An Example Hierarchy Where a Node Has a Maximum of *k* Children

Compared with the traditional single level layout as on Sprint, RELT displays a multi-level layout on one screen and thus extends the viewing scope. This benefits the user on two perspectives during navigation. First, by showing multiple levels, a RELT layout records a part of the navigation path (while traditional layout like Sprint shows only a point on the path). This helps users to maintain their mental map during navigation. Second, RELT shows more nodes on one screen. Take the *k-ary* hierarchy as an example; up to (k3 -1)/2 can be shown on one screen. This reduces the potential of selecting wrong nodes and also reduces the number of nodes leading to the target node.

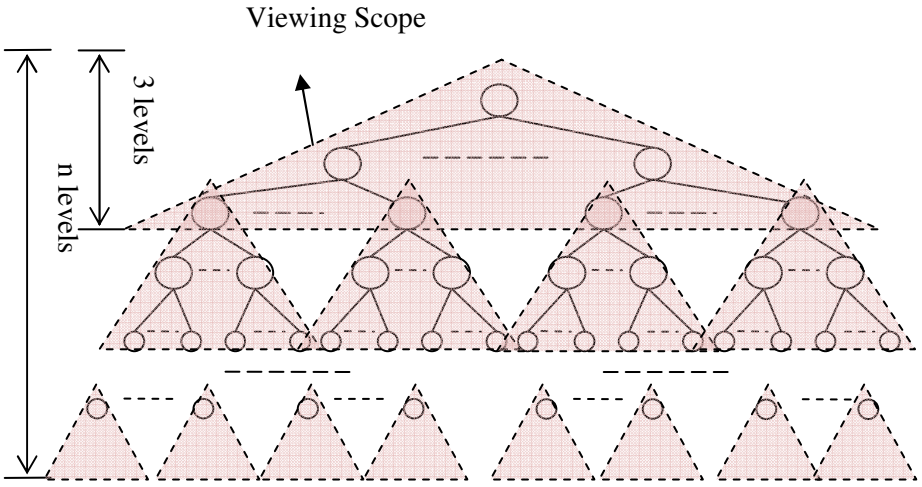


Figure 3.44. Observation Window Illustration

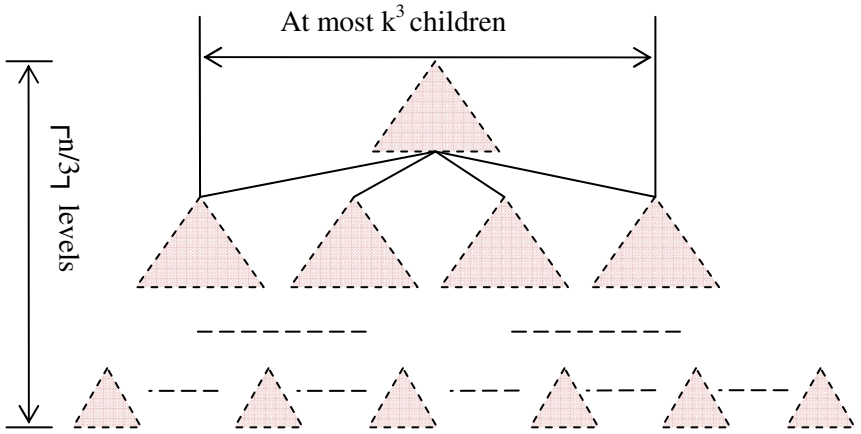


Figure 3.45. Hierarchy Reduction

RELT offers enhanced navigation ability by reducing the size of the hierarchy. As Figure 3.45 illustrates, the tree in Figure 3.43 shrinks to ⌈n/3⌉ levels of a k3-ary tree. The hierarchy size reduction is achieved by increasing the amount of information on the screen that the user can see at one glance.

Table 3.3 illustrates the number of touches, with a touch-screen interface, on the intermediate nodes that the user has to traverse to find the target node in the hierarchy. Table 3.4 shows the number of clicks for a button-based user interface. Assume that the target node is located at the ith level and there are totally j nodes in the first *i* levels. In the tables, "Explicit" means that the user knows where the target node is located. "Implicit" includes the worst and best cases when the user has no idea about the target node's location. The navigation approaches of Table 3.3 and Table 3.4 are touch-based and button-based, respectively.

Table 3.3. Number of Touches on a Touch Screen Interface

# of Screen Touches	Explicit	Implicit (worst)	Implicit (best)
Sprint	*i*-1	*j*-1	*i*-1
RELT	0	0	0

Table 3.3 shows that, using Sprint to navigate, the user has to touch *i*-1 times (i.e., traverse *i*-1 intermediate nodes) to arrive at the target node in the best case. However, in RELT, the user needs only one touch to arrive at the target (i.e., traverse 0 intermediate node).

Table 3.4. Number of Button Pushes on a Button-based Interface

# of Button Pushes	Explicit	Implicit (worst)	Implicit (best)
Sprint	*i*-1	*j*-1	*i*-1
RELT	*i*-1	*i*-1	*i*-1

Table 3.4 shows that instead of jumping from the root node to the target node as on a touch screen, the user has to push buttons to arrive at the target node level by level. Using RELT, the user has to traverse *i*-1 intermediate nodes. Cascading menus work equally well for both interfaces in the "Explicit" and best "Implicit" cases.

3.5.3.2 Empirical Study

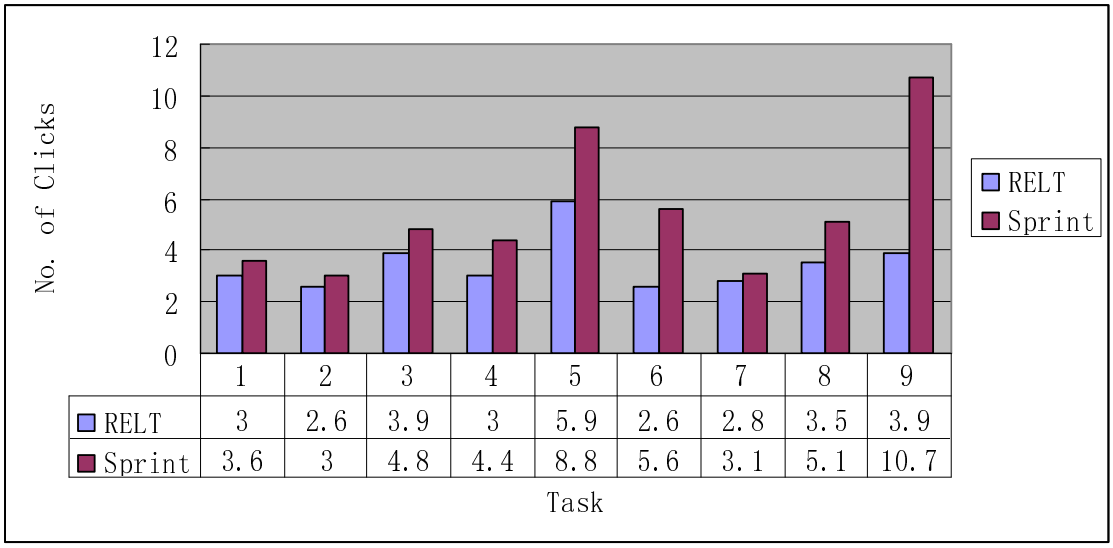
To make a fair comparison, we have also implemented the Sprint interface on a computer screen with the same size as that of the RELT interface. Interactions with the interfaces are through mouse clicks. We conducted an empirical study on the performance of each interface on the given example of hierarchical information (i.e., Sprint cell phone functions).

Eighteen Computer Science graduate students were involved in this study as the subjects, of whom half performed first on Sprint and then the RELT interface and the other half performed on RELT followed by Sprint. The subjects were asked to perform the following nine different actions on both interfaces:

1. Find if you called Mike on March 23;
2. Find whether David has called you on March 10;
3. Find the Settings of the Receiver Volume of the Speaker;
4. Go to Stopwatch Lap2;

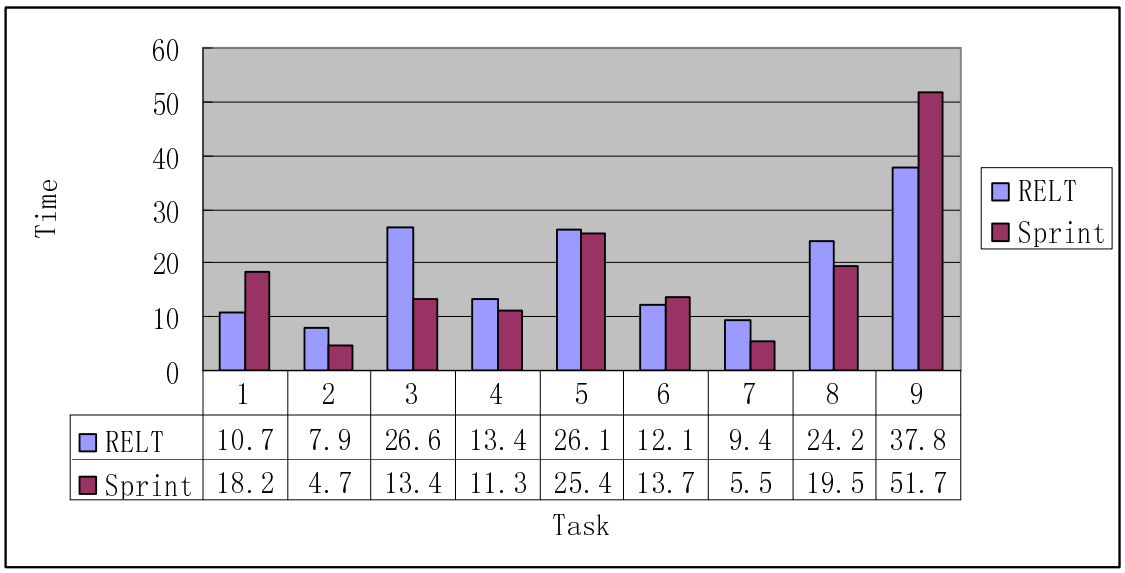
5. Find the Settings of the Power-off Tone;
6. Launch Tetris game;
7. View the Picture Album;
8. Find the Settings of the Messaging Signature; and
9. Find the Directory Services.

A touch-screen interface was assumed for both Sprint and RELT; the user could touch directly on the screen to select desired functions without pushing any buttons. Each "touch" was simulated by a mouse click on the selected node. We compared the number of clicks (touches) on the screen performed and total time in seconds taken for each action on both Sprint and RELT interfaces, as shown in Figure 3.46 and Figure 3.47, respectively. An action is considered complete after the node for the corresponding action was touched. The numbers in the figures for both RELT and Sprint are averaged over the 18 subjects.



	1	2	3	4	5	6	7	8	9
RELT	3	2.6	3.9	3	5.9	2.6	2.8	3.5	3.9
Sprint	3.6	3	4.8	4.4	8.8	5.6	3.1	5.1	10.7

Figure 3.46. Number of Touches (as Clicks) Performed



	1	2	3	4	5	6	7	8	9
RELT	10.7	7.9	26.6	13.4	26.1	12.1	9.4	24.2	37.8
Sprint	18.2	4.7	13.4	11.3	25.4	13.7	5.5	19.5	51.7

Figure 3.47. Total Time (in Sec) Taken

In Figure 3.46, RELT shows superior performance over Sprint with 0.3 to 6.8 fewer touches on average for the nine actions. Figure 3.47 shows mixed results on the speed of performing the nine actions. Although on average six actions (versus three) were completed in shorter times on Sprint than on RELT, the total averaged time difference is very close: 18.16 (Sprint) vs 18.89 (RELT). We observed that those who performed faster on Sprint had been using a similar cell phone interface and thus were fairly familiar with the Sprint interface (as evidenced below), while the RELT interface was totally unfamiliar to all the subjects.

We also asked the subjects several general questions in a questionnaire. When asked "In your experience in using cell phones, finding a piece of information on the current cell phone is …", six subjects chose "easy", seven chose "OK but improvements are needed", two with no response, and three felt unsatisfactory. Seventeen subjects responded positively to the question "Have you used the provided Sprint interface before, or are you familiar with this particular interface?" with only one negative. In response to the question "If you have to choose one of the two user interfaces, which one you will choose?" ten subjects reported to choose RELT, seven chose Sprint, and one did not indicate any preference.

CHAPTER 4

INFOSHAPE: VISUALIZE OVERVIEW OF MULTI-DIMENSIONAL INFORMATION

4.1 Introduction

The rapidly growing volume of multi-dimensional information provided by various data collecting applications, instruments, and especially the internet, requires techniques for meaningful interpretations and visual representations. Simple infovis techniques, such as line graphs and scatter plots, have been widely used for decades. With the help of line graphs or plotted points, viewers can easily understand one-dimensional information such as a function of one variable. Similarly, three-dimensional line graphs and scatter plots can describe the relationships among three variables. When the number of variables is four or even five, animation techniques and virtual environments may be used to convey essential multi-variable relationships. For relationships beyond five variables, standard geometric projection techniques are ineffective and the human perception system ceases to help.

Modern multi-dimensional infovis techniques involve encoding dimensionalities in graphical elements. Many multi-dimensional infovis techniques, such as parallel coordinates [59][60], can theoretically visualize multi-dimensional information with a huge dimensionality. In the early years, infovis served mostly, if not only, to convey results of statistical computation and data mining algorithms. Over the last decade, it has been extended to the fields of data processing, human-information interaction, and data management. Most previous graphical representations intend to express the detailed features of the information under study. Few offer graphical evaluation and comparison of information at a high yet intuitive level.

The idea is that if a multi-dimensional information set satisfies a given set of criteria, it will be represented as a perfect sphere. Otherwise, the distortions on the sphere indicate defective contents whose information does not satisfy some of the criteria. This idea has been filed in a patent [144].

The rest of this chapter is organized as follows: we first introduce some classic multi-dimensional infovis techniques followed by several types of classification which help in achieving a comprehensive view of multi-dimensional infovis researches. Second, we apply 5DITS to solve the problem: visualize overview of multi-dimensional information.

4.2 Classic Multi-Dimensional Infovis Techniques

4.2.1 Scatter Plot Matrices

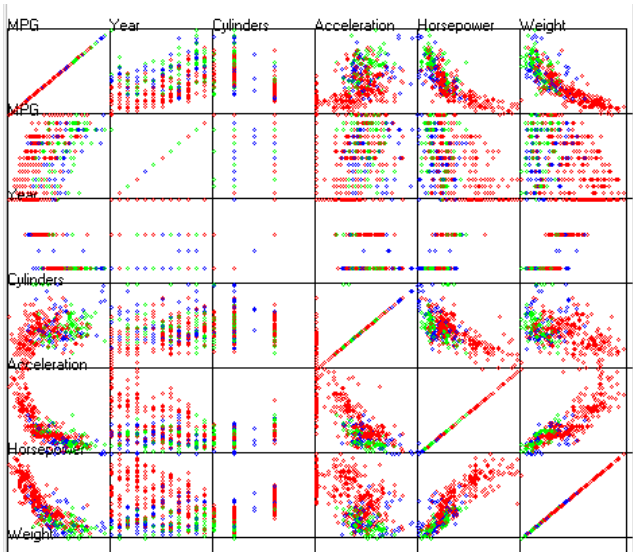


Figure 4.1. Scatter Plot Matrix of the Car Dataset [58]

Scatter plot is one of the most widely used infovis methods to express the relationship between two dimensions. Scatter plot matrices extend the scatter plot to higher dimensions. An *N* by *N* matrix visualizes an *N* dimensional dataset; each array of the matrix provides visualization of one

dimension versus every other dimension. Users can easily understand the interactions and relations between a two-dimension pair. The scatter plot matrix shown in Figure 4.1 which is published by Hoffman [58] visualizes a car dataset. This car data set concerns six attributes of cars which were manufactured in America, Japan and Europe from 1970 to 1982. Although the type of the car (the country that manufactures it) is also treated as an attribute, it is encoded by color (American cars are red, Japanese green, and European blue).

4.2.2 Parallel Coordinates

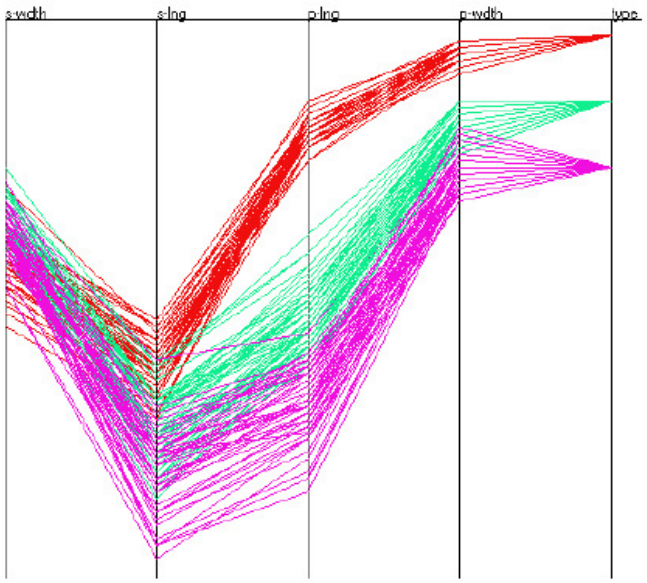


Figure 4.2. Parallel Coordinates of the Iris Dataset [58]

Parallel coordinates [59] [60] uses parallel axes instead of perpendicular axes to visualize multi-dimensional information. Each vertical line is used for the projection of a dimension or an attribute. The maximum and minimum values of a dimension or attribute are scaled to the upper and lower boundaries on the vertical line. An *n*-dimensional data is visualized as *n-1* lines which connect to each vertical line at the right dimensional value. Figure 4.2 [58] is the parallel coordinates visualization of an Iris dataset which is the physical measurements from three types

of flowers from Fischer 1936. Five attributes are concerned for the dataset: *s*-wdth, *s*-lng, *p*-lng, *p*-wdth and type. Instead of visualizing as an independent vertical line, type is also encoded by color.

4.2.3 Star Coordinates

Star coordinates was proposed by Kandogan [66] to visualize a multi-dimensional data set. The coordinates are located on a circle emanating from the same origin at the center. An *n*-dimensional datum is visualized as a single point and each dimension is treated uniformly at the cost of coarse representation. Star coordinates assists users to gain a quick insight into underlying data, especially on the cluster discovery and multifactor analysis.

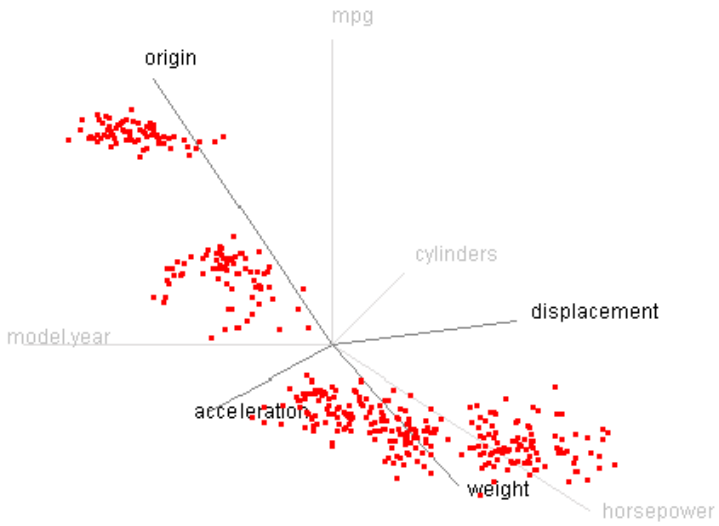


Figure 4.3. Visualization of Star Coordinates on a Car Dataset [66].

Figure 4.3 illustrates the star coordinates of a car dataset. This data set concerns the mpg, cylinders, horsepower, acceleration, displacement, original, year, etc., of approximately 400 cars manufactured worldwide. After applying navigation functions (scaling, rotating, turning off certain coordinates), users can easily find that there are four clusters in the data as shown in Figure 4.3.

4.2.4 VisDB

VisDB [68] aims at supporting the query specification process by each pixel which denotes an attribute of a record in the database. Users are guided by the graphical representation which arranges and colors the pixel according to the relevance of the records with respect to the query. By querying the data base, users not only get the record fulfilling the query, but also records approximately fulfilling the query.

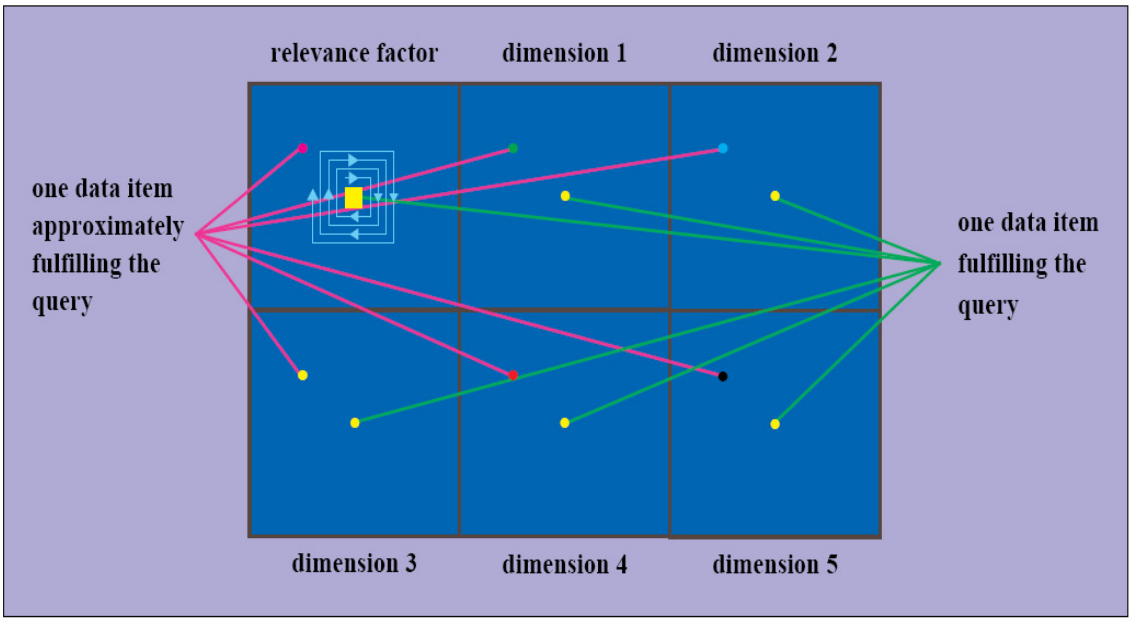


Figure 4.4. Arrangement of Windows for Displaying Five-dimensional Data [68]

To connect the perception of the overall result and individual dimensions, VisDB generates a separate window for each dimension of the query and arranges them in the left-right manner shown in Figure 4.4. In the overall window, the pixels are positioned according to the computed factors in a spiral manner. In a separate window, the pixels are located in the same relative position as the overall result in the overall window. By relating different windows, users can understand underlying data. Figure 4.5 shows an eight-dimensional data set having 7,000 data items.

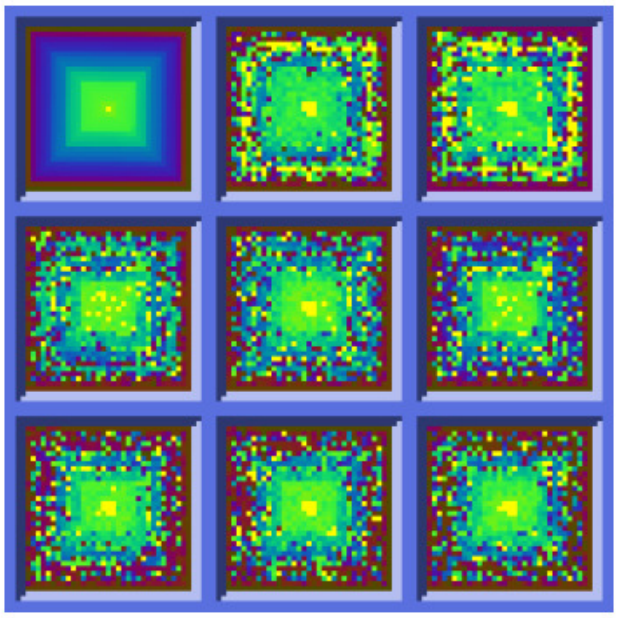


Figure 4.5. Visualization of Eight Dimensional Data in VisDB[68]

4.3 Classification of Multi-Dimensional Infovis Techniques

Due to the lack of formal description of the infovis space, current infovis design tends to be ad-hoc and experience-based, much like an artist's painting. Researchers have tried to solve this problem by categorizing infovis techniques into groups. Understanding the similarities and differences of different groups will guide and facilitate the infovis design process. This section reviews four notable taxonomies of current multi-dimensional visualizations.

4.3.1 Taxonomy by Keim et al.

As in Table 4.1, Keim [69][67] and Kriegel contribute a taxonomy in which multi-dimensional infovis techniques are categorized into six groups: geometric projection, icon-based, pixel-oriented, hierarchical, graph-based and hybrid. The metrics of categorization tend to be on the visual presentations, that is, how viewers observe and understand these graphical presentations.

For more detailed discussions and a comparison of most of the listed visualization techniques refer to [27] [68] [69] [94] [132].

Table 4.1. Visualization Techniques Taxonomy by Keim *et al.*

Categories	Visualization Techniques	Characteristics
Geometric Projection	Scatter plots [2][30] Landscapes [133] Prosecution Views [41][117] Hyperslice [128] Parallel Coordinates [59][60]	Show data projections and geometric transformations
Icon-based	Chernoff Faces [30][122] Stick Figures [95][96] Shape.Coding [14] Color Icons [68][79]	Visualize data as icons
Pixel-oriented	Recursive Pattern Technique [74] Circle Segment Technique 0 Spiral- and Axes Techniques [68]	1. Each attribute value of every data record is assigned a pixel. 2. Attributes of each data record are presented in different sub windows.
Hierarchical	Dimensional Stacking [78] World-within-World [41] Treemaps [113] Cone Trees [108] InfoCube [104]	Visualize data using hierarchically partitioned subspaces
Graph-based	Basic Graph (Straight-Line, Polyline, Curved-Line) [12]	Convey the meaning and relationship of underlying data by graph
Hybrid	Any combination from above	

4.3.2 Taxonomy by Card et al.

Data-oriented taxonomy attempts to categorize infovis techniques by data domains or data types that are compatible with these techniques. Card *et al.* [30] provide a representative data-oriented taxonomy in which infovis techniques are initially grouped into four levels. Infovis tools in the first level assist users to visually access the information external to their immediate environment. Tools on second level support users to quickly execute tasks within their workspace by interactive graphical representations. Visual Knowledge Tools (VKT) on the third level depicts

graphical representation of data and associating interaction. Fourth-level infovis tools mainly focus on producing an intuitive view of data which has potential visual form.

VKT encompasses most infovis techniques targeted at data tables and is further classified according to "Visual Structure (VS)." The concept of VS specifies how space encodes information, that is, the dimensionality of the data graphical representation used. VKT is classified by the VS that it can adopt. In some sense, this reflects the kind of task that VKT can support and complete. Typical types of visual structure are: Physical; 1D, 2D, 3D; Multi-D; Tree and Network. Such classification is constructed based on the OLIVE (The Online Library of Information Visualization Environments), assembled by students in Shneiderman's class [149] and Shneiderman [116]. OLIVE divides information visualization techniques by eight visual data types: temporal, 1D, 2D, 3D, multi-D, Tree, Network and Workspace.

4.3.3 Taxonomy by Chi et al.

Data State Reference Model is another notable taxonomy [27][30] which breaks down infovis techniques not only by the data types, but also by the processing operations. As illustrated in Figure 4.6, the Data State Reference Model consists of data stages (black diamonds) and data transformation operators (gray rectangles). Data stage specifies the nature of data being operated on and the data type in different data stages differs. There are four data stages-value, analytical, abstraction, visualization abstraction and view-which contain raw data, meta-data, visualizable data, and viewable data, respectively. Data transformation operators and within stage operators are defined, depending on if the operator data transforms the data from one stage to another. Transforming essential data structure from one data stage to another requires one of the three data transformation operators: data transformation, visualization transformation and visual mapping transformation. Within stage operators do not change the data structure they operate

upon, that is, the data will stay in the same data stage after being applied to a within stage operator.

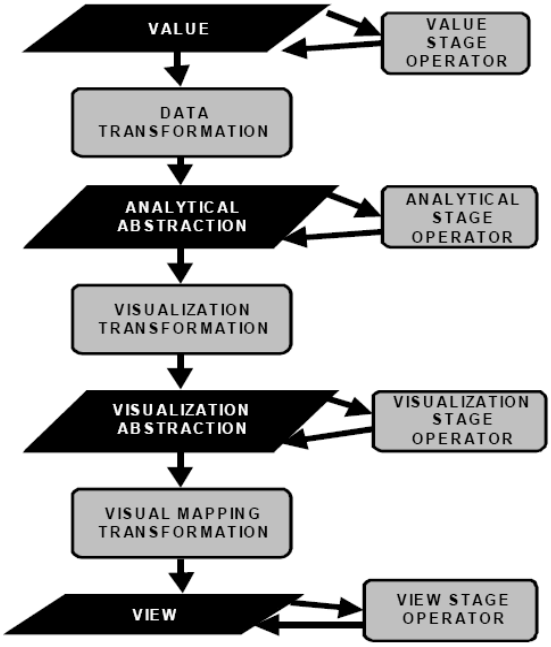


Figure 4.6. Information Visualization Data State Reference Model [27]

4.3.4 Taxonomy by Valiati et al.

Argued as "When developing an infovis technique, the analytic and exploratory tasks that a user might need or want to perform on the data should guide the choice of the visual and interaction metaphors implemented by the technique [123]," Valiati *et al.* regard infovis technique design as a user-task driven process and classify infovis techniques based on user-tasks.

One point that requires attention is that most researchers do not regard the taxonomy based on data type and that based on user-task as conflicting taxonomy methods. Valiati *et al.* [123] think both the represented data and user tasks should guide the design, selection and the development of an infovis technique. Shneiderman combines the data type (1-dimensianol, 2-dimensional, 3-

dimensional, temporal, multi-dimensional, tree and network) and the user tasks (overview, zoom, filter, details-on-demand, relate, history and extract) to classify visualization techniques [116].

In summary, previous taxonomies categorize infovis techniques based on various point of metrics, reflecting different views of the infovis research. The taxonomy of Keim *et al.* views the collection of visualization techniques according to the attribute-visual feature mapping and the output graphical representations. Such classification probably matches human perception of visualization the best, so it may be the best way to briefly survey the current infovis techniques for a novice. This taxonomy, however, neither guides users to select the appropriate infovis techniques for a particular task for certain data type nor assists engineers to understand the process of infovis technique implementation. Contrasted with little guidance for developers of Keim's taxonomy, Chi *et al.* provide a clear roadmap by breaking down the whole visualization pipeline into four data stages, linking by data transformation operators. Taxonomy methods of Card *et al*. and Valiati are based on the data types and user tasks, respectively, two of which are considered as major elements to drive the design, development and evaluation of visualization.

4.4 Applying 5DITS to Design InfoShape

Multi-dimensional visualization, which creates visual presentations in more than one dimension, helps users to gain an insight into represented information from different perspectives. This section uses the 5DITS to design two infovis techniques for visualizing the overview of multi-dimensional information. Record InfoShape (RInfoShape) and Dimension InfoShape (DInfoShape) are proposed to visualize multi-dimensional information set as a 3D sphere whose appearance denotes how much the provided information satisfies a pre-defined set of criteria. By comparing the shapes of different multiple information sets, overall content similarities and

differences can be quickly captured. RInfoShape and DInfoShape are evaluated by case studies on a Java program and comparison of US life tables.

4.4.1 Define the Triggering Problem

In either daily life or research work, we frequently meet situations where multi-dimensional information comparisons are needed. The first type of comparison has explicit criteria to which information needs to compare. For instance, we usually encounter questions such as "Are the data collected by sensors in the acceptable range?", or "Do the new trucks satisfy customers' needs?" and "How does the quality of life in the USA compare with average level of the world?" Table 4.2 lists the multi-dimensional information and their associated criteria of these three example questions.

Table 4.2. Example Questions and Associated Explicit Criteria

Multi-dimensional Information	Explicit Criteria
Data collected by sensors	Acceptable Range
New trucks information	Customers' needs
Quality of life in USA	Average level of the word

The second type of comparison, however, does not have any explicit criteria. These comparisons may be like: "What's the difference of consumption behavior between people in City A and people in City B?" The second type comparisons can be converted to the first type by simply taking one side, for example, the people in City B, as the criterion. Then, the above question is transformed to "How do people in City A perform differently compared to the people in City B?" All these comparisons can be generalized as: "How to quickly understand and compare multi-dimensional information under a set of criteria," which requires an overview of the complex information to be captured at a glance. Therefore, we define the triggering problem as: "Visualize Overview of Multi-dimensional Information"

4.4.2 Decompose the Triggering Problem

The step of decomposing the triggering problem in this case is fairly simple. We consider the triggering problem, "visualize overview of multi-dimensional information," as an undivided problem. The logic tree thus has the simplest form: having only one node – the *TP* root. Although the triggering problem cannot be further divided, it essentially contains two meanings. The first meaning is "quick view". This requires that the difference between information sets must be encoded into a visual feature needing the minimum users' cognitive efforts. Users can directly tell the difference with a glance. The second is "high-level" view. The expected infovis technique should aim at providing a high-level overview which somewhat ignores the individual datum.

4.4.3 Design a Temporary Infovis Technique

Few of the aforementioned multi-dimensional infovis techniques are suitable to serve as a temporary infovis technique. The most important reason for this unsuitability is that metaphors of these techniques are naturally designed to show certain features of multi-dimensional information sets, rather than comparing them. Although it is true that separately showing features of information sets can lead to a comparison, the lack of criteria inevitably results in requiring additional cognitive efforts. In addition, showing features of an individual datum trumps illustrating the features of the whole information set. Here a new idea is proposed so that the shape of a 3D sphere can be used to illustrate the difference of multi-dimensional information sets on a high level.

4.4.3.1 Use Shape of 3D Sphere to Show Overivew of Multi-dimensional Information

As mentioned before, if a multi-dimensional information set satisfies a given set of criteria, it will be represented as a perfect 3D sphere. Otherwise, the distortions on the sphere indicate defective contents whose information does not satisfy some of the criteria. In summary, the shape of 3D spheres is used to show the overview of multi-dimensional information. Before going further, three questions need to be answered:

4.4.3.1.1 Why 3D?

Springmeyer *et al*. [119] show that 2D displays are usually used to convey precise relationships, whereas 3D displays are to gain a qualitative understanding and represent ideas. 2D displays beat 3D when tasks need to show detailed specifications, such as focused analysis, precise navigation, and distance measurement [116]. In contrast, 3D displays facilitate shape recognition, approximate navigation and position, and 3D space survey [65]. Pizlo [156] claimed that the only way to study the role of simplicity in shape perception is to use 3D visualization (or 2D images of 3D objects). In other words, 3D objects adapt to a human's perception system better than 2D objects.

Our objective is to present the similarities of global contents and highlight the differences among multi-dimensional information sets. The comparative assessment of multi-dimensional information sets is represented at a high level and local details can be viewed upon request.

4.4.3.1.2 Why Shapes?

Shape is a natural visual perception for human beings. Marr [82] argues the process of determining how a shape is interpreted as something "deep buried in our perception machinery." Inspired by economical paleolithic animal representational arts, Halverson [46] explored why the

visual dimensions in these arts are so limited and why these minimally represented arts are so recognizable. By pointing out that the profiles of animals are chosen in preference to other views, Halverson's research proved the importance of shapes in the perception process. Due to the prominent role of shapes, we use shape to represent the global view of multi-dimensional information.

4.4.3.1.3 Why Spheres?

A sphere as a 3D shape can be detected and identified easily due to its equidistance property that makes it simpler than other 3D objects. An experiment by Goethe [74] proves this: an afterimage of a square becomes more and more circular with the passage of time. Koffa [74] claimed this as one statement in Gestalt psychology that the observers' internal forces, which bias a percept toward simplicity, beat the external forces which are produced by the square because the square is not as simple as the "simplest form" - circle. According to Gestalt psychology, the equidistance property ensures the sphere as the "simplest form," and thereby easier to detect and identify than other 3D objects.

4.4.3.2 Format and Terminology

Multi-dimensional information needs to be pre-processed and formatted into a data structure before being translated into graphical representations. The author's research only takes the table format into account which was first introduced by Card *et al.* [30]. Multi-dimensional visualization on table format data is specifically termed Table Visualization [58]. A table is a structured data format organized by *M* rows and *N* columns. Defined by Card *et al*., the rows represent attributes and columns represent records. Specifically, we define the columns and the rows as the attributes and the records, respectively.

Wong and Bergeron [132] provide a profound discussion on the terminology where "dimensions" refer to independent variables, whereas "variates" refer to dependent variables. Usually, the data dependence relationship is the right task of data exploration and is unknown in advance. To avoid confusion, this dissertation uses "attribute" rather than dimension to denote both dependent and independent variables. "Data record" (record) is used to denote a tuple expressing the relationships among attributes. The specific content associated with a particular attribute of a record is defined as "attribute value." Based on the definition of attribute value, we term "attribute value arrange" denoting the range of attribute values of a particular attribute.

4.4.3.3 Record-based Visualization and Dimension-based Visualization

Record-based Visualization (RBV) presents the record-based view of a table. In an RBV graphical representation, a record is visualized as a visual object that encodes visual features, representing all the dimensions of the record. Radviz [58] in Figure 4.7 is an example of RBV, which conveys a car data set. This car data set contains six dimensions. These cars were manufactured in America, Japan and Europe from 1970 to 1982. The type of the car (the country that manufactures it) is also treated as a dimension, so a car data set actually contains seven dimensions. The type dimension is expressed by colors and the other six dimensions are denoted by their corresponding axes. A car record is denoted by an integrated visual object which is realized as a unique plotted point in Radviz. Each point conveys a car's type by a color and conveys other six dimensions by the projection of point's location on their axes. There is no strict definition of visual object. It can be a plotted point (radial coordinate [58]), an icon (Chernoff faces [30][122]), a polyline (parallel coordinates [59][60]), a polygon (circular parallel coordinates [58]), a pixel (hyperslice [128]) and any other possible forms. In summary, a visual object is widely accepted as a graphical unit which can independently contribute

underlying information. RBV graphical representations illustrate each record as a unit, so they emphasize the individuality of records.

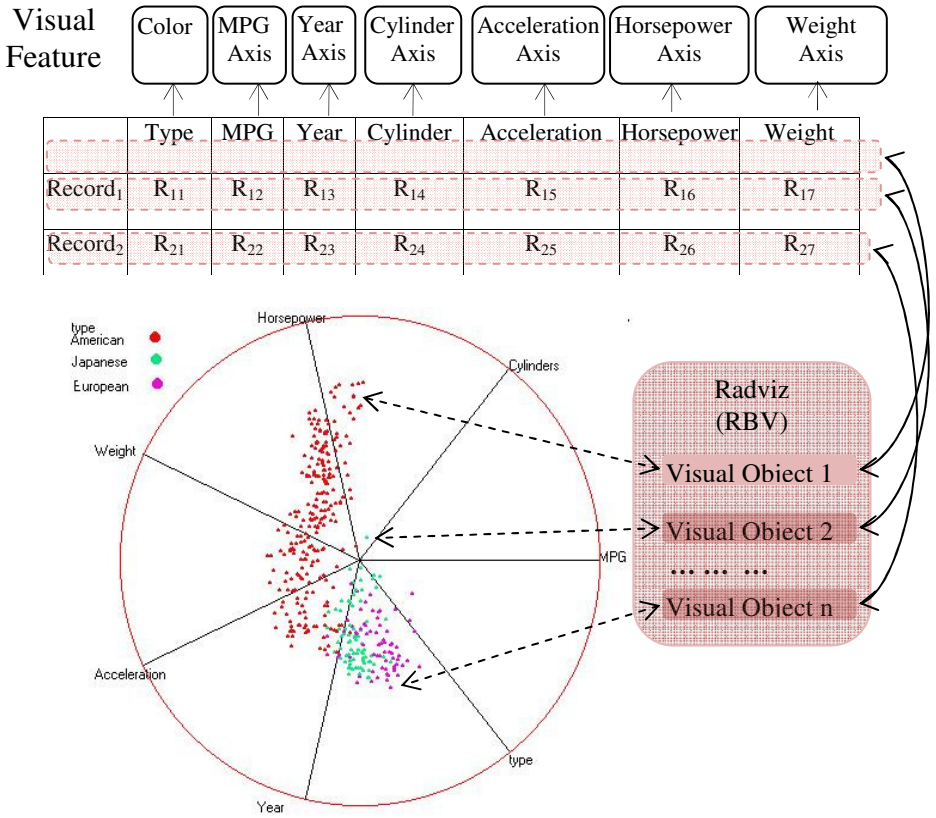


Figure 4.7. Radviz – An Example of RBV

Dimension-based Visualization (DBV) denotes the visualization which presents a table in the dimension-based view. DBV considers each dimension as an individual aspect of multi-dimensional information and visualizes each dimension as a visual object. Unlike the unsplit visual object in RBV, a visual object in DBV usually consists of sub visual objects, each of which denotes a dimension of a record. Therefore, in contrast to RBV, there are no single visual objects which can convey all dimensions of a record, i.e., dimension information is scattered into multiple visual objects.

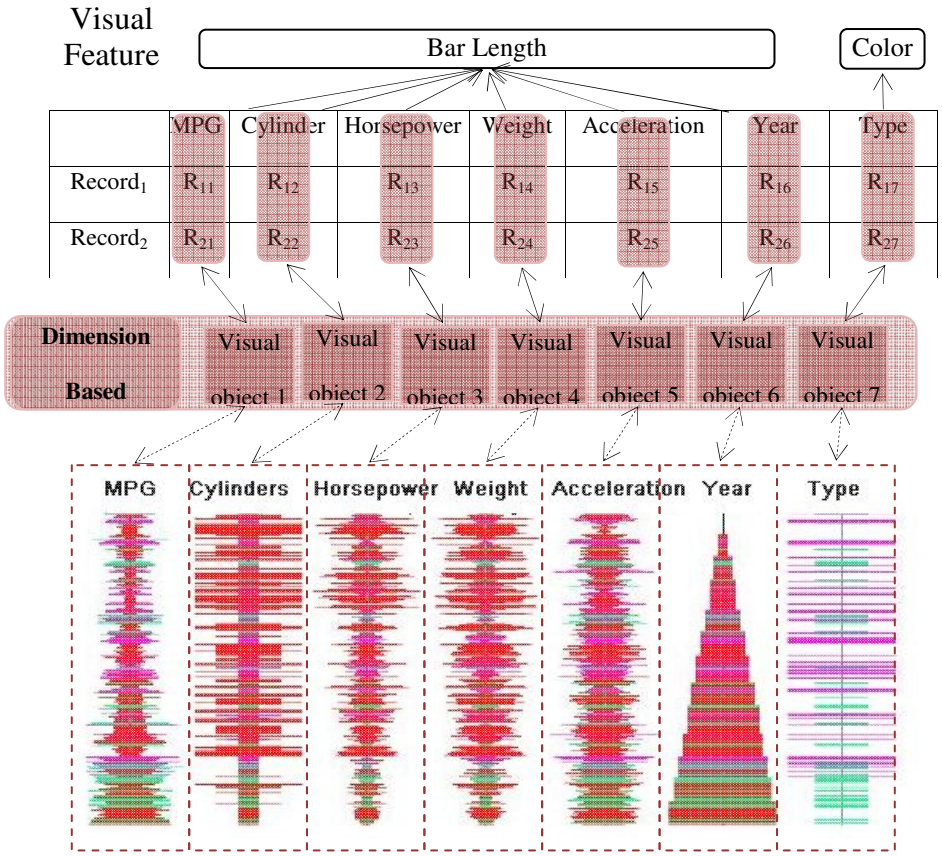


Figure 4.8. Survey Plot – An Example of Dimension-Based Visualization

Survey plot [58] technique in Figure 4.8, an example of DBV, visualizes the same car set as Figure 4.7 does. The type of a car is encoded by a color and the other six dimensions are all expressed by the length of a horizontal bar. Dimensions of a car data set are visualized as seven visual objects which are located in the left-right arranged sub windows. It's not as easy as RBV for a user to understand an individual record in a DBV representation. DBV, however, shows its advantage to convey a table at a higher level view. Users can gain a quick overview of a dimension or relationships between dimensions of the data.

One can distinguish RBV from DBV by simply determining "if all visual features are integrated into one visual object" or "if there is a unique visual object that can convey all dimensions of a record." If the answer is yes, it is RBV, otherwise it is DBV. All icon-based visualization

techniques in Keim [67] et al.'s taxonomy are considered the RBV. The pixel-based visualization techniques, showing visual features in different sub windows, belong to the DBV. Table 4.3 categorizes visualization techniques surveyed in [69][67] into RBV and DBV.

Table 4.3. Classification of Multi-dimensional Infovis into RBV and DBV

RBV	Parallel Coordinates [59][60] Circular Parallel Coordinates [58] Radial Coordinate Visualization [58] Star Coordinates [66] Landscapes [133] Chernoff Faces [30][122] Stick Figure [95][96] Color Icon [79][68] Shape Coding [14] Star Glyphs [120] Worlds-in-world [41] Dimensional Stacking [78] Treemap [113] Cone Trees [108] InfoCube [104]
DBV	Prosection Views [41][117] Hyperslice [128] Scatter Plot Matrices [2][30] Spiral & Axes techniques [68] Recursive Pattern [74] Circle Segments 0

4.4.4 Record InfoShape

A conceptual model of RInfoShape is first introduced and then developed to visualize a Java program under the compilation criteria.

4.4.4.1 Notations

$R = \{R_1, R_2 \ldots R_n\}$ is a set of records.

$A = \{A_1, A_2 \ldots A_n\}$ is a set of areas and A_i specifies the area of R_i on the sphere:

$$A_i \{[\alpha_{from}, \alpha_{to}], [\beta_{from}, \beta_{to}]\}$$

where $[\alpha_{from}, \alpha_{to}]$ and $[\beta_{from}, \beta_{to}]$ denote the longitude and latitude ranges on the sphere, respectively.

Area Distribution Function (ADF ()) calculates area set *A*.

$F = \{F^*, F_1, F_2 \ldots F_m\}$ is a set of surface functions. F^* is the normal sphere function. F_i can be any user-defined surface function.

$C = \{C_1, C_2 \ldots C_k\}$ is a pre-determined set of criteria.

$S = \{S^*, S_1, S_2 \ldots S_k\}$ is a set of information validity status. S^* denotes the valid status. Invalid status S_i indicates the violation of C_i.

$V = \{V_1, V_2 \ldots V_t\}$ is a user-defined set of visual features. A visual feature can be any visual clue used to encode information.

Ri $(a \in A, s \in S, f \in F, v \in V)$ denotes the ith record and *a, s, f, v* denote the area, validity status, surface function and visual feature, respectively.

4.4.4.2 Conceptual Model

The RInfoShape conceptual model in Figure 4.9 consists of scene and rendering models, mapping information into a graphical presentation in two steps.

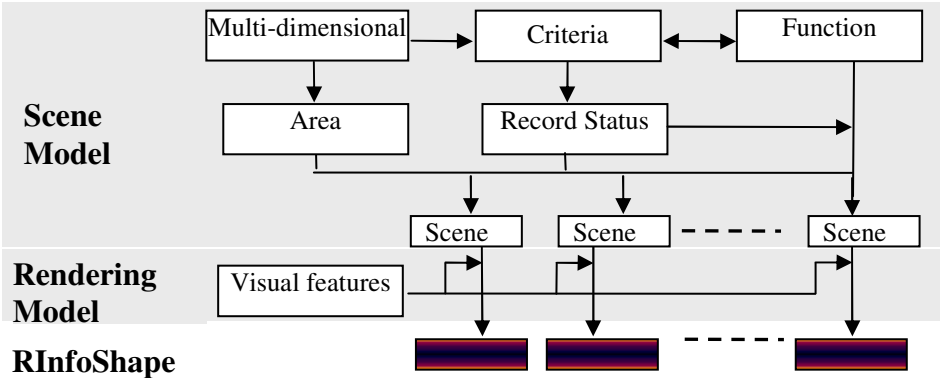


Figure 4.9. RInfoShape Conceptual Model

4.4.4.2.1 Scene Model

A scene model specifies two aspects of a record: where to locate the record on the sphere, and how to visualize the validity status of the record.

- **Area Distribution**

A record view associates each record with a visual object. To locate each visual object on the specific area of the sphere, RInfoShape uses an area distribution function, or *ADF(),* that returns a unique area for that visual object. *ADF()* could be defined based on different applications. Various functions can be developed to assign a unique area on the sphere to a record. For example, spherical helix function [154] can be used to divide a sphere into non-overlapping strips and each strip can be assigned to a record. A dimensional area distribution function maps two dimensions of a record into the longitude and latitude on the sphere. The two selected dimensions must uniquely identify a specific record so that the location on the sphere is unique.

- **Record Status**

The status of a record denotes the record's validity, determined by examining the record against all related criteria. If a record satisfies all of its associated criteria, it is assigned a valid status S^*; otherwise, the record is assigned S_i ($S_i \in S$) which denotes the violation of C_i (for simplicity of presentation, C_i here represents a criterion or a subset of criteria in C)

- **Function Assignment**

During function assignment, each record is assigned a surface function according to the record status. F^* is assigned to the record with a valid status. F_i ($F_i \in F$) is assigned to the record whose status is S_i.

4.4.4.2.2 Rendering Model

The rendering model assists users to identify the records' validity by adding visual features (such as color and texture). It allows users to define visual features for individual visual objects that fit their own cognition, as shown in the Figure 4.10.

```
V = {V1, V2 … Vn} // Initial Visual feature set
While (Continue editing V) {//User-defined visual feature set
     Add a visual feature OR Delete a visual feature from V
}
for (index = 1; index <= n, index ++){
   Rindex.v = Vi  // User assign a visual feature to the record.
}
```

Figure 4.10. Rendering Model

4.4.4.3 Case Study: Evaluating a Java Program

This section applies RInfoShape to a Java program. The graphical representation assists users in evaluating the example program under compilation criteria. The program was submitted by a student in a Java course as a project.

4.4.4.3.1. Pre-processing

The constructed program should follow predetermined syntactical and semantic criteria. The example program contains five classes, namely *Person, Sponsor, Student, Friend,* and *Car*, each containing several methods. For simplicity, only syntactical requirements are considered as criteria. The formal definition of the criteria is given as

$$C = \{C_i \mid C_i \text{ is a syntax criterion, } i = 1, 2, 3\ldots\}.$$

A quick overview of the example program under the criteria can reveal how well the program passes the compilation process.

Table 4.4. Compilation Results of the Example Java Program

Class Name	Method Name	Validity Status	Violated Criteria
Car (C = 1)	chooseModel (M =1)	InV	End statements with ';'
Car (C = 1)	firstPayment (M =2)	InV	End statements with ';'
Car (C = 1)	chooseColor (M =3)	InV	End statements with ';'
Car (C = 1)	chooseYear (M =4)	InV	End statements with ';'
Car (C = 1)	choosePrice (M =5)	InV	End statements with ';'
Car (C = 1)	chooseDealer (M =6)	InV	End statements with ';'
Friend (C =2)	checkSchedule	V	None
Friend (C = 2)	missOrNot (M =2)	V	None
Friend (C = 2)	hug (M =3)	V	None
Person (C = 3)	setName (M =1)	InV	Use a class after defining it
Person (C = 3)	count (M =2)	InV	Use a class after defining it
Person (C = 3)	speak (M =3)	V	None
Person (C = 3)	SetAge (M =4)	V	None
Sponsor (C = 4)	payApartment (M =1)	InV	Use a method after defining it
Sponsor (C = 4)	payCar (M =2)	InV	Use a method after defining it
Student (C = 5)	findFriend (M =1)	V	None
Student (C = 5)	addFriend (M =2)	V	None
Student (C = 5)	countFriend (M =3)	V	None
Student (C = 5)	inviteFriend (M =4)	V	None
Student (C = 5)	buyCar (M =5)	V	None

Table 4.4 summarizes of the evaluation results of the example code. Each record describes a compilation result of a method in a class. Dimensions "Class" and "Method" together uniquely specify a method (no method overloading in the program). "C" and "M" are assigned to each method to uniquely identify it. "C" represents the alphabetical order of a class name and "M" represents the order of the method's appearance in the class. The record set can be defined as:

$$R = \{R_{(c,m)} \mid R_{(c,m)} \text{ is a record in compilation description table}\}.$$

The "V" in "Validity Status" denotes that the record satisfies all the criteria; "InV" denotes "Invalid status". Dimension "Violated Criteria" is self explained. Figure 4.11 illustrates a piece of code, corresponding to the first record in Table 4.4.

```
…
Class Person {
                                        Typo: it should be noticeChange.
  public void setName (){
    …
    noticechange("name has been updated");
  }
  public void noticeChange (String msg){
    System.out.println(msg);
  }
  …
}
```

Figure 4.11. A Code Example

4.4.4.3.2. Scene Model

- **Area Distribution**

Area Distribution Function - *ADF()* determines the area occupied by each record on the surface of the sphere. This dissertation uses a dimensional area distribution function due to its two properties. First, dimensional *ADF()* returns a unique area for each record. Second, dimensional *ADF()* aims at mapping two information properties to longitude and latitude of the sphere, which matches a human's perception of a sphere.

C and *M* are applied in the dimensional *ADF()*. The area of $R_{(c,m)}$ can be obtained using Equation 4.1, where C_{NO} denotes the number of classes and M_{NO} denotes the number of methods contained in the class *c*.

$$a = ADF(c,m) = \begin{cases} [2\pi * c / C_{NO}, 2\pi * (c+1) / C_{NO}] \\ [\pi * m / M_{MO}, \pi * (m+1) / M_{NO}] \end{cases}$$

Equation 4.1. Area Distribution Function

Equation 4.1 maps the class and method indices to the longitude and latitude, respectively. This mapping ensures two desirable features. First, the spaces on the same longitude belong to the methods in the same class. Second, geographical distance to the North Pole denotes the order of

appearance in that class. Figure 4.12 illustrates the locations of four methods belonging to one class.

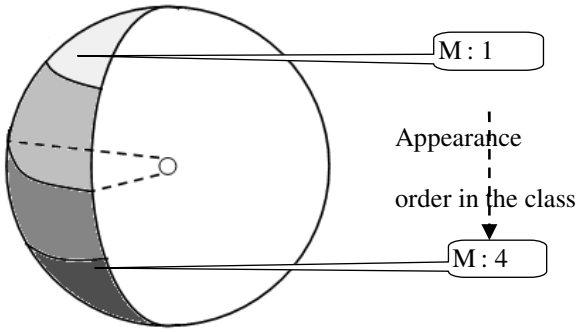
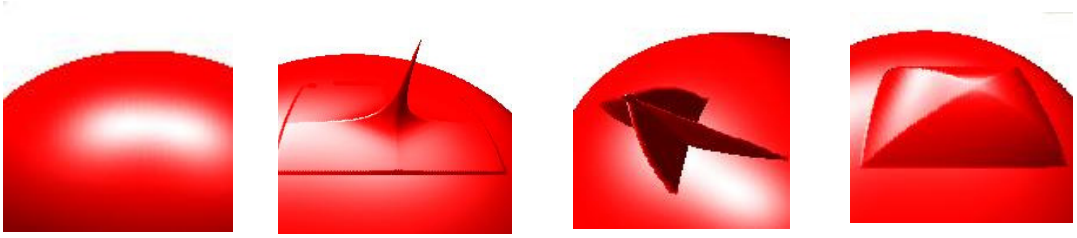


Figure 4.12. Area Distribution using Dimensional ADF().

- **Function Assignment**

Figure 4.13 illustrates four functions used. Function *F** can only be assigned to valid methods. Tine, cross, and hunch are functions assigned by users to the specific violation criteria.



(a) *F** Sphere Function (b) Tine Function (c) Cross Function (d) Hunch Function

Figure 4.13. Four Example Surface Functions

In this case study, we assign the Tine function to the criterion "Use a method after defining it," Cross function to "Use a class after defining it" and Hunch function to "End statements with ';'." Following these function assignments, the example program's evaluation by compilation can be visualized in Figure 4.14.

A spike on the sphere indicates that the method assigned to that area violates certain criteria. The shape of the spike illustrates the corresponding criterion violated.

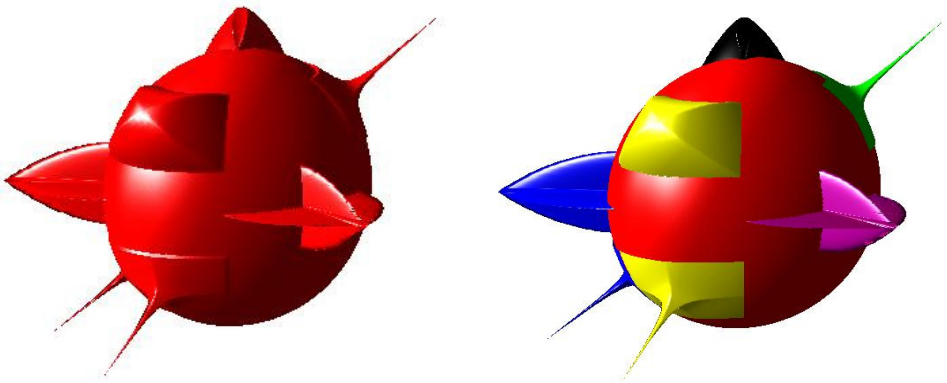


Figure 4.14. RInfoShape of the Example Program

4.4.4.3.3. Rendering Model

As Figure 4.14 (a) shows, our method intuitively assists users to understand the program under the pre-defined compilation criteria. Users, however, may want to extract more information about criteria violation from the graphic perspective. In the rendering model, users have many options to define visual features to facilitate their understanding.

We take the visual feature *V* which consists of only color as an example; other visual features can be easily added.

Color is used to distinguish the invalid methods among different classes. In other words, we map the dimension "Class Name" to colors. For instance, this dissertation uses blue, yellow, purple, green, and black to illustrate the Car, Friend, Person, Sponsor, and Student respectively. Figure 4.14 (b) illustrates the visualization of the example program after colors are applied.

4.4.5 Dimension InfoShape

DInfoShape is designed to provide the aforementioned dimension view. In contrast to RInfoShape, DInfoShape does not assign each record a unique area on the sphere where all visual features are integrated; instead, it assigns a unique area to each dimension. There are two

steps in constructing a DInfoShape. We first arrange the multi-dimensional information on a 2D panel, and then map the 2D plane to a 3D sphere.

4.4.5.1 2D Arrangement

The 2D information arrangement in DInfoShape is adapted from that of VisDB [68].

4.4.5.1.1 VisDB 2D Arrangement

VisDB [68] aims at supporting querying specification by graphic representations. It visualizes the resulting data which match or approximately match the query. This is potentially similar to our task: "visually evaluate multi-dimensional information under a set of criteria," that is, providing a visual impression on how approximately the multi-dimensional information matches the criteria.

In VisDB, to relate the visualization of an overall result to that of the individual dimensions, the display window is divided into several sub-windows which are arranged next to each other. Usually, the upper-left sub-window is for the overall visualization, and the other sub-windows visualize individual dimensions.

As shown in Figure 4.15, VisDB adopts the spiral-shaped arrangement. The distance to the spiral center denotes the approximation or how much the associated information matches the criteria. The approximation of a record (in overall sub-window in Figure 4.15 (a)) and the individual dimension of that record (in individual dimension window in Figure 4.15 (b)) are represented by a relevance factor (see appendix for calculating the relevance factor). Data with the highest relevance factors are centered in the middle of the overall window. The smaller the relevance factors are, the closer the corresponding data are positioned to the edges of the overall window.

Moreover, in the dimension windows, data are placed at the same relative position as they are in the overall window.

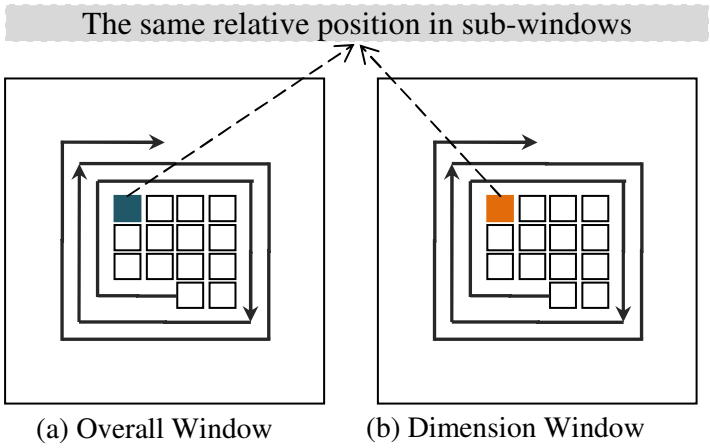


Figure 4.15. Original Pixel Arrangement in VisDB

4.4.5.1.2. DInfoShape 2D Arrangement

In VisDB the position for each record in the dimension window is determined by its position in the overall window; the distance to the center in individual dimension windows does not reflect the approximation of data in that dimension,

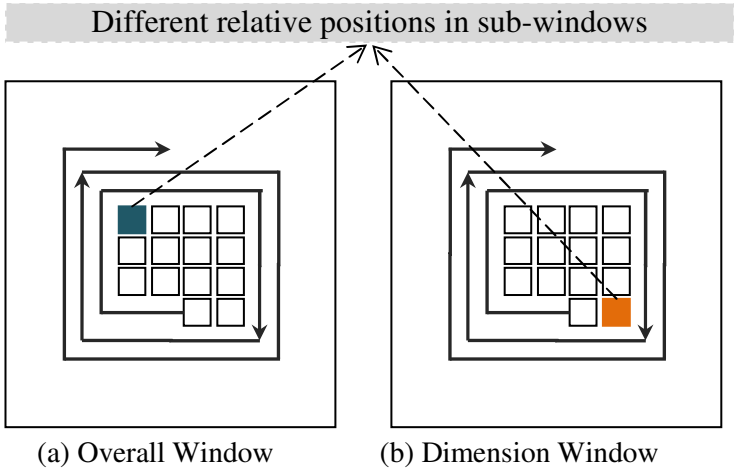
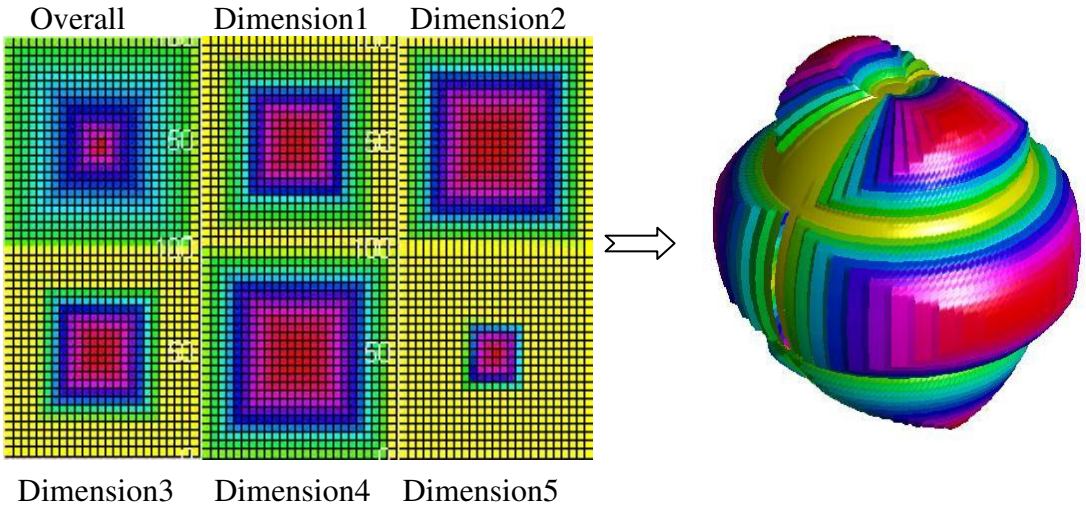


Figure 4.16. Pixel Arrangement in DInfoShape

DInfoShape maintains the consistency of spiral properties in both overall and dimension windows. In the overall window, DInfoShape arranges data in the descending order of relevance factors. The data in the center of overall window denote those violating certain criteria. The closer the pixels' positions to the edges of the window, the better the records match the criteria. Similarly, in dimension windows, data are arranged based on the distance of that dimensional attributes of records in descending order (see appendix for distance calculation).

As illustrated in Figure 4.16, the same data in the overall window and individual dimension windows may have different positions. This modified arrangement clearly visualizes the distance of the data to the criteria in the spiral layout in each window, thereby helping to evaluate either the overall or a single attribute of that data. Furthermore, InfoShape takes 3D shape as a visual cue, which delivers information much quicker than maintaining the same relative positions for corresponding data (as in VisDB).



(a) 2D Arrangement (b) DInfoShape

Figure 4.17. DInfoShape of a Randomly Generated 5-dimensional Information Set

DInfoShape uses the similar color scheme as that in VisDB, ranging from yellow to green, cyan to blue, and magenta to red, to denote the approximation with respect to the criteria in descending order.

Figure 4.17 (a) shows the 2D arrangement of randomly generated 5-dimension information set with 639 records. The upper-left sub-window is the overall window and the other five sub-windows are the dimension windows. In this example, data in each dimension ranges from the minimum dimension value to the half of the maximum value in that dimension.

4.4.5.2 From 2D to 3D

We use the inverse function of Mercator projection [147] to convert a 2D panel into a 3D sphere. The inverse function of Mercator projection in Equation 4.2 can uniquely map a pixel to a point on a sphere.

$$\varphi = 2\tan^{-1}(e^{y}) - \frac{\pi}{2} = 2\tan^{-1}(\sinh(y)) \qquad \lambda = x + \lambda_0$$

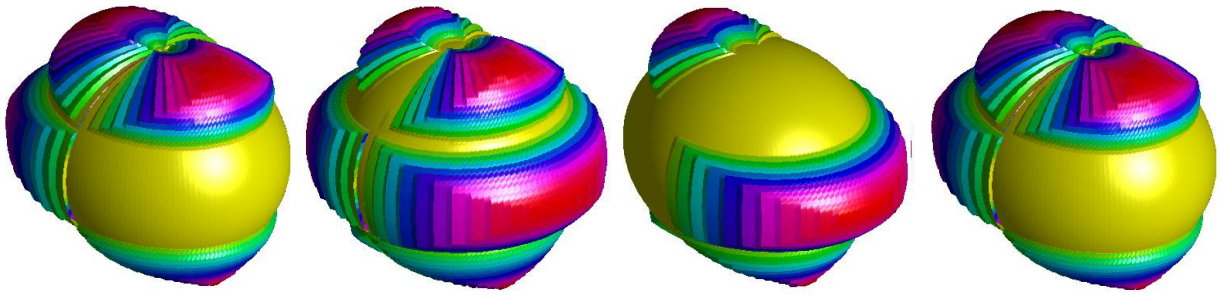
Equation 4.2 Mercator Projection

where x and y denote the 2D coordinates of the pixel, φ and λ denote the longitude and latitude, respectively.

The distance of a point to the sphere center equals the relevance factor (pixel in overall window) or distance (pixel in dimension window). Figure 4.17 (b) illustrates the 3D visualization of Figure 4.17 (a).

Figure 4.18 visualizes four different 5-dimensional information sets. Each dimension is associated with a criterion, set from the minimum value to the half of the maximum value of that dimension. From a quick glance at these spheres, one can notice the similarity of Figure 4.18 (a)

and Figure 4.18 (d). This means that the corresponding sets are about the same with respect to the criteria.



(a) (b) (c) (d)

Figure 4.18. Dimension InfoShape of 5 Different 5-dimensional Information Sets

Table 4.5 illustrates the complexity of InfoShape which depends on two algorithms in the two steps. The first algorithm in the step of locating information on 2D plane has a function for computing distance and relevance factor, and a quicksort function. The complexity of these two functions is O(n) and $O(n^2)$, respectively. The complexity of algorithm for mapping 2D to 3D sphere is O(n). Therefore, the complexity of InfoShape is $O(n^2)$.

Table 4.5. Complexity of InfoShape

	Algorithm	Functions	Complexity
InfoShape	Algorithm for locating information on 2D plane	Compute distance and relevance factor	O(n)
		Sort according to distance and relevance factor	$O(n^2)$
	Algorithm for mapping 2D plane to 3D sphere	Reverse function of Mercator projection	O(n)

4.4.5.3 Case Study: USA Life Table

This section applies DInfoShape to United States life tables [10]. The data source of the life tables includes the death statistics of US residents during 1999-2001, counts of US resident

births during 1997-2001, population estimates based on the 2000 decennial census, population and death counts from the Medicare program during 1999-2001. The visualization provides an intuitive comparison of mortalities of different races or different genders of a population.

Period life tables represent expected life status of a hypothetical cohort if it experienced throughout its entire life certain mortality conditions. These mortality conditions are constructed according to collected data of a fixed and relatively short duration. For example, according to the 1999-2001 life tables [10], the life expectancy for an American was estimated at 76.83 years. This life expectation is the average age of death for all Americans born in 1999 who are assumed to experience the age-specific death rates prevailing for the actual population in 1999-2001. A period life table can be regarded as a "sketch" of current mortality rates that prevailed at a particular time. Visual comparisons among period life tables reveal the age-specific death rates of different subdivisions of a population and can significantly help demographical analysis.

We apply DInfoShape on five life tables which contain data for the total, males, females, white, and black populations in United States [10]. There are six dimensions in each life table:

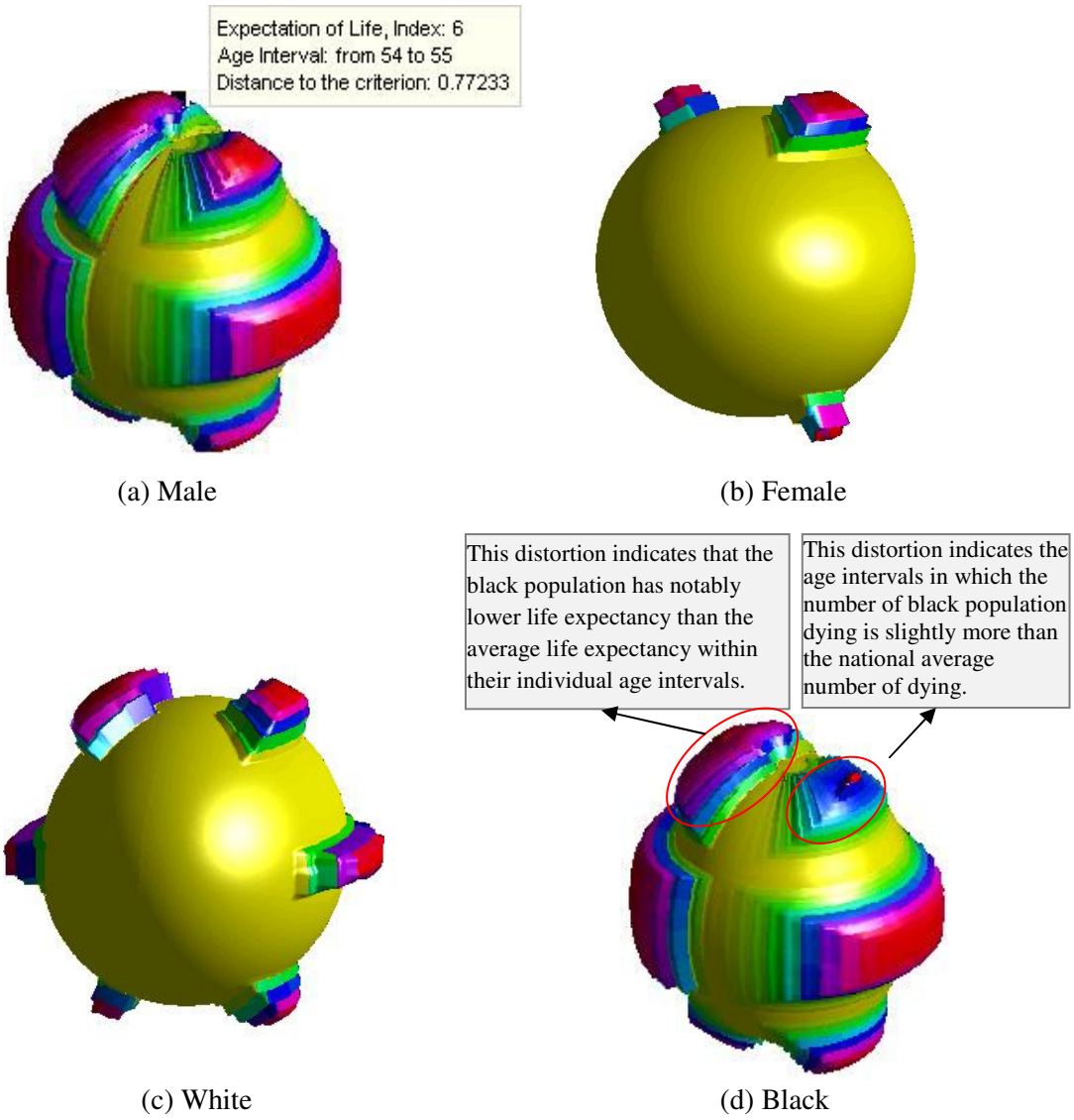
- Probability of dying (*nqx*) measures at the age from *x* to *n+x* on the basis of mortality rates of 1999-2001. For example, the probability of dying between 0 and 1 year old is 0.00696 in the life table for total US population, namely, 6.96 out of 1,000 American babies died before reaching their first birth days.
- Number of surviving (*lx*) at age *x*, which is the beginning of each age interval, is calculated by applying nqx to the survivors of the original cohort at the beginning of each age interval. According to the life table for total US population, 99,305 out of 100,000 American babies successfully passed their first birth days.
- Number of dying (*ndx*) within an age interval from *x* to *n+x* is calculated by multiplying nqx

and *lx*. For example, 695 out of 100,000 American babies died before reaching their first birth days.

- Person-years lived (*nLx*) illustrates the number of person-years lived by the table cohort within an age interval from *x* to *n+x*. Therefore, 99,436 in 0-1 age interval is the total number of years lived by the original 100,000 American babies.
- Total number of person-years lived (*Tx*) illustrates the total number of person-years of alive persons since the beginning of an age interval from *x* to *n+x*. For example, the figure 7,584,011 is the total number of years lived by 99,305 American babies after reaching their first birth days.
- Expectation of life (*ex*) is the expected average remaining number of years of the survivors who have entered the age x. It is obtained by dividing *Tx* with *lx*. According to the life table of total population, the average expectation of an American is 76.83.

We visually compare two life tables. A reference life table includes a set of references. Each figure in the reference life table represents a criterion of a dimension in a specific age interval. A data life table is used to compare with the reference life table. If the data in this table equal those in the reference life table, we set the distance to zero. If the examined data do not satisfy the criterion, the distance will be the absolute numerical difference. Therefore, if the examined life table exactly satisfies the reference life table, the comparison will be visualized as a perfect sphere; otherwise, it is visualized as a sphere with distortions representing the numerical differences. Users can rotate the sphere and drill down by clicking its surface to view details. The clicked point on the surface is toggled and detailed information of this point is shown in a data tip window (Figure 4.19 (a)). The data tip window by default provides information of the

selected data, such as dimension, age interval, and the distance to the reference data. Users can customize the information in the data tip window.



(a) Male (b) Female

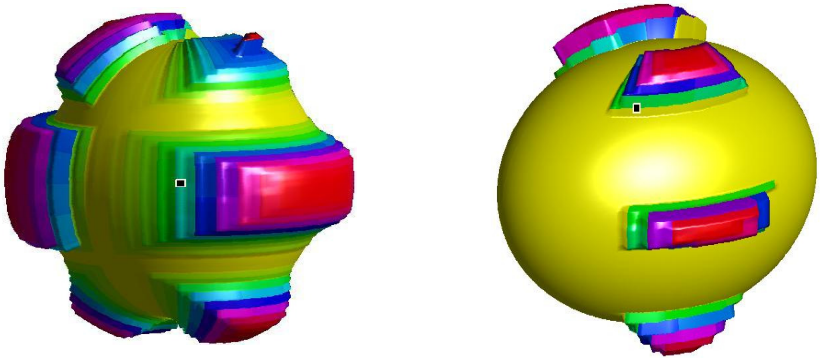
(c) White (d) Black

Figure 4.19. DInfoShape Examples for the Life Tables for Four Groups of American People

Figure 4.19 visually compares the life tables of male, female, white and black populations with the life table of the total US population. The sphere for female (Figure 4.19 (b)) appears to have the least distortions, implying that the female population lives with less probability of death, compared to other three groups. The white population (Figure 4.19 (c)) lives with a higher probability of death than the female population, but lower than male and black populations.

According to Figure 4.19 (a) and (d), the probabilities of death of male and black populations are almost the same. The size of the distortion indicates the amount of difference of the data from the reference. The bigger the distortion is, the more significant the difference is. For example, as illustrated in Figure 4.19 (d), the distortion of number of surviving is much more than that of number of dying, indicating that the number of surviving is significantly more than the number of dying.

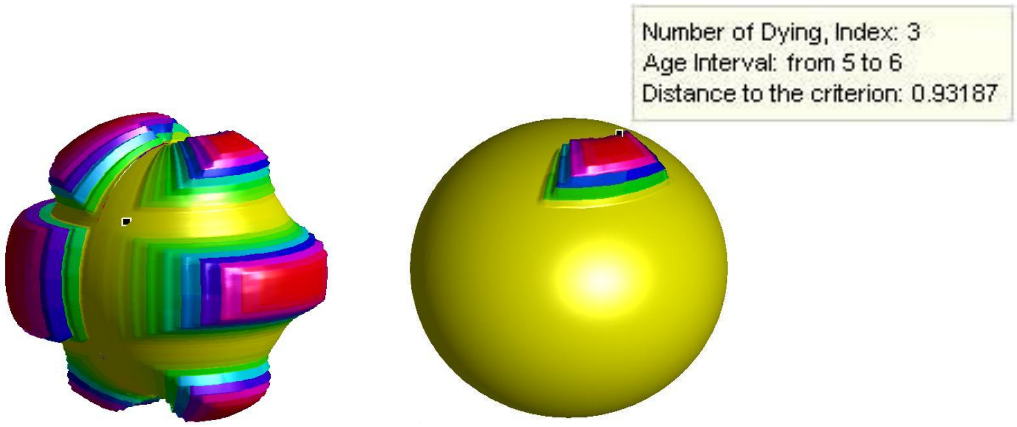
Figure 4.20 compares the life tables of white and black populations. Figure 4.20 (a) sets the life table for the white population as the reference and that for the black population as the examined data. Figure 4.20 (b) visualizes comparisons of the reference and examined data reversed. The fact that Figure 4.20 (b) has less distortion than Figure 4.20 (a) indicates that white population lives with a less probability of death than the black population.



(a) (b)

Figure 4.20. Comparisons between the Life Tables of Black and White Populations

Figure 4.21compares the life tables of male and female populations. The massive distortions on Figure 4.21 (a) indicate that the male population life expectation is lower than that of the female population. From Figure 4.21 (b), we can observe that, except the "Number of Dying" dimension, all death data for the female population meet the criterion in the male life table.



(a) (b)

Figure 4.21. Comparisons between the Life Table of Male and Female Populations.

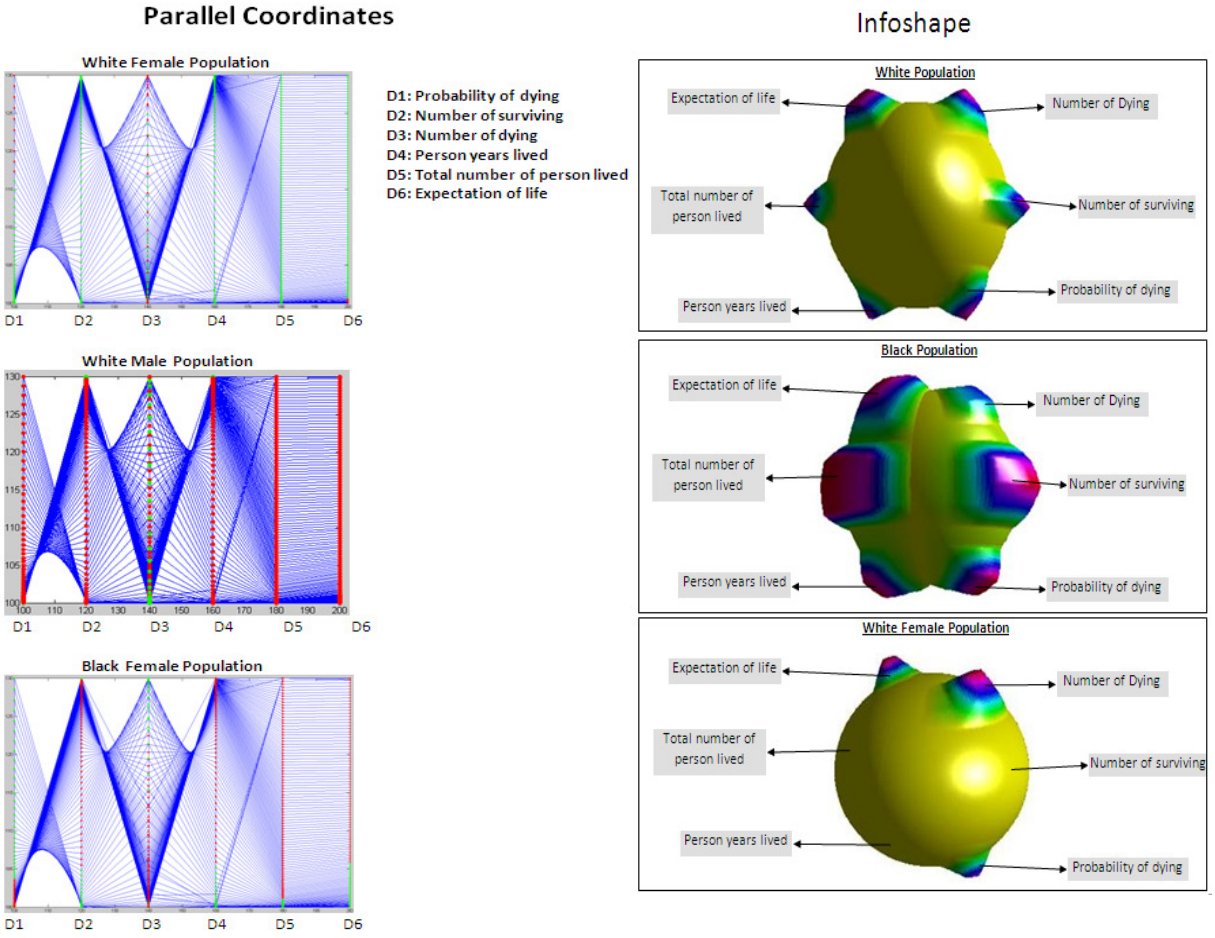
4.4.6 Evaluation of InfoShape

InfoShape aims at providing users a quick way of viewing multi-dimesional information at a high level. The general idea is to visualize multi-dimensional information as a 3D sphere whose appearance denotes how much the provided information satisfies a pre-defined set of criteria. By comparing the shapes of different multiple information sets, global content similarities and differences can be quickly captured. The underlying multi-dimensional data in this dissertation is formed in table format. This dissertation first introduces a new classification of multi-dimensional visualization. This classification divides multi-dimensional visualization into RBV and DBV, which lead to RInfoShape and DInfoShape respectively. RInfoShape visualizes each record as an individual visual object. If a record satisfies all criteria, the associated visual object is a smooth surface of a sphere; otherwise, the visual object is visualized as a spike. The visualized sphere consists of many individual visual objects; therefore users have to make an extra cognitive effort to understand features of the entire information set. *DoS* of RInfoShape is assigned 70 percent. On the other hand, DInfoShape visualizes each dimension of an information

set on a part of a continuous surface and hence provides a high level view. *DoS* of DInfoShape is assigned 90 percent due to the 10 percent deduction for the lack of interaction.

4.4.7 Empirical Study

An empirical study was conducted to assess the real performance of InfoShape. InfoShape and parallel coordinates are used to show the life tables of five different groups of American people.



(a) Parallel Coordinates (b) InfoShape

Figure 4.22. Example Snapshots of Empirical Study

InfoShape is applied on white, black and white female people, and parallel coordinates technique is applied on white female, white male and black female. The average life status of all American people is set as criteria. In the parallel coordinates, color green indicates that the associated data

is valid; otherwise, the color red denotes defective data. Sixteen students in four departments (six in computer science, four in electrical engineering, four in school of management, and two in material engineering) are involved in this evaluation test. All of these students are neither familiar with InfoShape nor familiar with parallel coordinates.

The empirical study is conducted using repeated measures. Above mentioned sixteen students are equally divided into two groups. The number one group of student first answered the questions for parallel coordinates and then answered those for infoshape; the number two group of students finished the questions in an opposite way. The questions for two techniques are designed in the same manner: the first two questions asked users to choose the worst group if only one dimension is concerned; the third and forth question asked users to compare two different dimensions; the fifth and sixth question asked users to rank the groups of people according to a particular dimension, the last one asked users to rank groups of people due to the overall situation. The questions are list as below.

Questions for parallel coordinates:

1. Which group has the worst situation only regarding the dimension: probability of dying?

2. Which group has the worst situation only regarding the dimension: number of dying?

3. For white male population, is the dimension number of dying has a better situation than the dimension of person years lived?

4. For white female population, is the dimension probability of dying has a better situation than the dimension of number of dying?

5. Sort the three groups according to the situation of dimension probability of dying.

6. Sort the three groups according to the situation of dimension number of dying.

7. Sort the three groups according to the overall situation.

Questions for InfoShape:

1. Which group has the worst situation only regarding the dimension: probability of dying?

2. Which group has the worst situation only regarding the dimension: number of dying?

3. For white population, is the dimension number of dying has a better situation than the dimension person years lived?

4. For black population, is the dimension probability of dying has a better situation than the dimension number of dying?

5. Sort the three groups according to the situation of dimension probability of dying.

6. Sort the three groups according to the situation of dimension Total number of person lived.

7. Sort the three groups according to the overall situation.

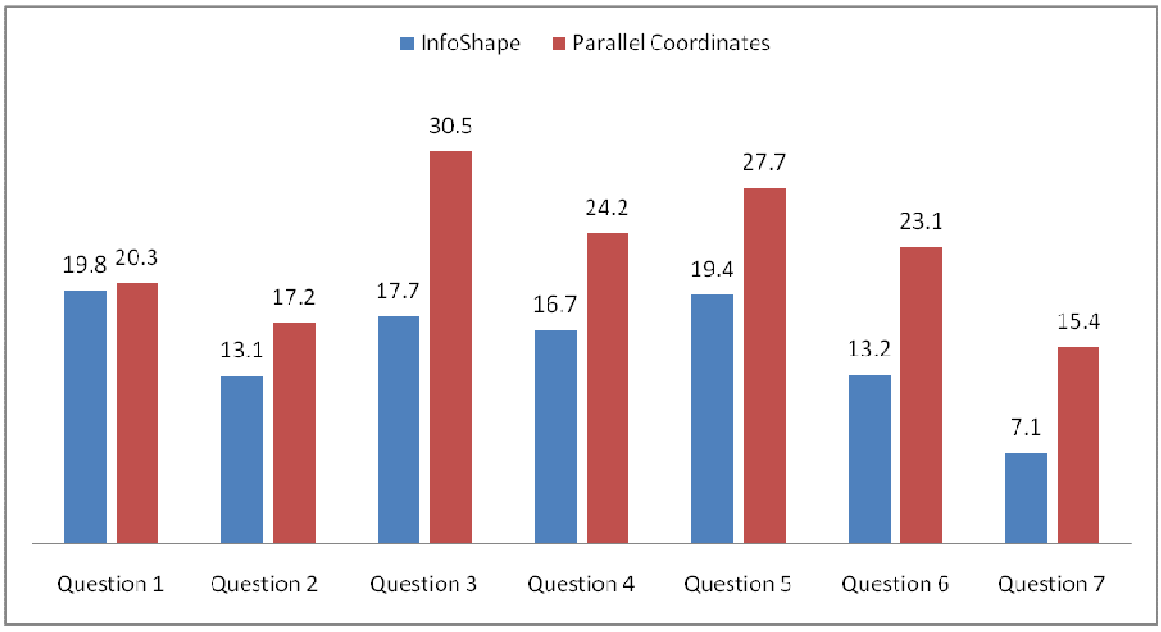


Figure 4.23. Comparison of Average Time Taken (in sec.)

The amount of time used for each question is recorded. Although the precision of answers when users used these two techniques is approximately the same, there is big difference of time taken when using two techniques. Figure 4.23 illustrates the comparison of average time taken for each

question. As shown in the picture, users decided quicker using InfoShape than they did using parallel coordinates.

CHAPTER 5

SCITREND: VISUALIZE POPULARITY TRENDS OF COMPUTER SCIENCE RESEARCH AREAS AND RELATIONSHIPS

5.1 Introduction

Computer science is the study of theoretical foundations of information and computation, and of practical techniques for their implementation and application in computer systems [137].

Over the past few decades, computer science has rapidly expanded into a very complicated structure. More and more new research areas and topics have been proposed. Current computer science has a wide range of research areas, from theoretical areas such as algorithm analysis, limitations of computation, to more practical ones, such as computer language, hardware and software programming. The Computer Science Accreditation Board (CSAB) identifies four crucial areas of computer science: theory of computation, algorithms and data structure, programming methodology and languages, and computer elements and architectures. In addition, the following areas are also identified as individual important areas: software engineering, artificial intelligence, computer networking and communication, database systems, parallel computation, distributed computation, computer-human interaction, computer graphics, operating systems, and numerical and symbolic computation. Identifying and defining a computer science research area is also a research topic which is mainly about methodologies for domain analysis and modeling. The research work in this area is numerous but is peripheral to the research in this dissertation. The above description introduces computer science in a module

that dissects the whole discipline into separate areas which seem quite unconnected. However, the reality is just the opposite: most of these research areas are heavily related. First, connections exist among them. Although each area has its own history, if one has a closer view, most areas either branch from their parent research areas or co-exist with some highly related research areas, rather than existing absolutely independently. Second, a research area also has inner connections which are usually in the form of citations between papers. Therefore, approaching a research area in isolation may result in an unbalanced view or even misunderstanding. Actually, awareness of these connections becomes necessary for people whose work is associated with computer science. Newcomers to a discipline can follow citations through papers, thereby achieving a quick overview of a topic without too much deviation. By portraying relationships among research areas, computer science educators provide students a better understanding of this discipline than teaching computer science as a simple combination of unrelated research areas. Computer science researchers use connections among different research areas to identify emerging new interdisciplinary areas, and use citations to stay abreast of new ideas. Conference organizers sometimes apply connections to classify submitted papers, and paper reviewers use a paper's citations to decide its originality.

The research in this dissertation focuses on creating a visualization tool that helps people understand the computer science research areas' popularity trends and connections, including ones both among and within areas. The popularity of a computer science research area represents the state of being paid more attention and widely researched.

The typical method is citation analysis which is the examination of frequency, patterns and graphs of citations in articles and books. [42][110][135]. The popularity (usually also defined as "impact" or "quality") of an article is assessed by the number of times that it is mentioned in

other research articles. Similarly, the popularity of a computer science research area is assessed by the number of times that all research articles in this area have been mentioned. In other words, the popularity of an area is the sum of popularities of papers in this research area. Citations are usually seen as the underlying connections between papers. Having a large number of citations shows the connections among research articles, or even associating research areas.

A traditional bibliographical database offers users resourceful citation data, and also supports functions such as searching, sorting, filtering and so on. Such existing databases include: IEEE Xplore [140], the ACM Digital Library [31], and CiteSeer [45] [136].

5.2 Related Work

Citation analysis mainly involves two types of citations. Before discussing the related work of these two types of citation analysis, a brief notation of terminology is made. When an article references another article or an author references an author's work, a relationship is created between articles or authors. Let's take article-article citation for example. If a research article *A* has a reference to an article *B*, we say *A* references *B*. The inverse form is called a citation, i.e., B cites A. This choice of terminology is consistent with that used in the ACM Digital Library. However, the terms "citation" and "reference" have been conflated for a long time. For example, CiteSeerX [45][136] uses "citation" instead of "reference". In this dissertation, if *A* has a reference to *B*, we simply say *A* is the citing article, and *B* is the cited article.

The first type of citation refers to an author-author connection when an author mentions another author's (possibly the same person) research in his/her work. Author-author citation is usually used in author co-citation analysis (ACA). The basic idea of co-citation is that if *A* and *B* are both cited by a *C*, then they are said to have certain relationship, even though they are not directly related. If *A* and *B* are both cited by a group (*D*) of items, they may have a tight

relationship. The more items in *D*, the stronger the relationship between *A* and *B* is. Co-citation analysis aims at showing literature's coherence and topic (or research area) similarities. ACA maps research works and the people who produce them. The author pairs are extracted from raw data. The number of citations between a pair of authors is then accounted for regardless of which of their research works are cited. Visualizing an author-author citation network mostly yields an implicit social network between the involved researchers. Therefore, it potentially identifies the "monarch" authors and their subordinates.

Another type is article-article citation which emphasizes research work regardless of the authors who produce them. An article-article citation network consists of bibliographical articles representing research works. Each article has a collection of references to other articles. Therefore, an article-article citation network is a directed graph where each node is a research article, outgoing edges point to cited papers and incoming edges come from citing articles. A large amount of research has been done on analysis of the article-article citation network, most of them focusing on showing the influence of certain research works and relationships between several research fields. The research in this dissertation focuses on visualizing the article-article citation network. In the rest of this dissertation, when we mention citation, we are referring to an article-article citation.

5.2.1 Citation Network Analysis

This section introduces some visualization tools for citation network analysis.

5.2.1.1 CiteWiz

The motivation of CiteWiz [34] starts from two limitations of typical ways of visualizing a citation network as a simple node-link diagram. The first limitation is the poor scalability of

node-link diagrams to dense networks; the second is that additional aggregation methods are needed to reduce density to achieve acceptable readability. Moreover, these two limitations result in another shortcoming: citation chains are hidden in the messy node-link layout.

Assuming that a researcher has different roles when interacting with a citation visualization tool, instead of using an established classification CiteWiz builds its own taxonomy on the concept of users' roles and associated goals and tasks. Six pre-selected active researchers are involved in the process of building this taxonomy. More details of defining the taxonomy can be found in [34][35]. In addition, another hallmark of CiteWiz is to interpret a citation as a causal relation. This interpretation is straightforward: for two papers connected by a citation, first, the cited paper was produced before the citing paper, second, the citation indicates that the author of the citing paper has read and been influenced by the cited paper. The term "influence" is then used to represent causality: if *A* cites *B*, *A* is influenced by *B*.

CiteWiz provides three visualizations: timeline visualization for overviews, influence visualization for detailed views, and an interactive concept map for exploring keywords and co-authorship.

Timeline Visualization:

Figure 5.1 illustrates the timeline visualization in CiteWiz metaphorizing Newton's famous statement: "If I have seen further, it is by standing on the shoulders of Giants." Each icon represents a research work or an author whose size is scaled proportionally to the number of citations the research work or an author has received. This visualization builds a vertical timeline running from bottom to top (from 1995 to 2002). This timeline is vertically divided into suitable units which are assigned space according to the largest icon located in associated time units. The

horizontal arrangement of icons is computed by a greedy algorithm which aims at maximally reducing the distance to the centerline of the diagram.

The diagram in Figure 5.1 illustrates the timeline visualization of the IEEE Information Visualization (InfoVis) conference in terms of authors. The large and black icons at the bottom refer to the authors of the paper: "Visualizing the No-Visual: Spatial Analysis and Interaction with Information from Text Documents" [129]. This paper was published at the first InfoVis conference in 1995 and received the highest number of citations.

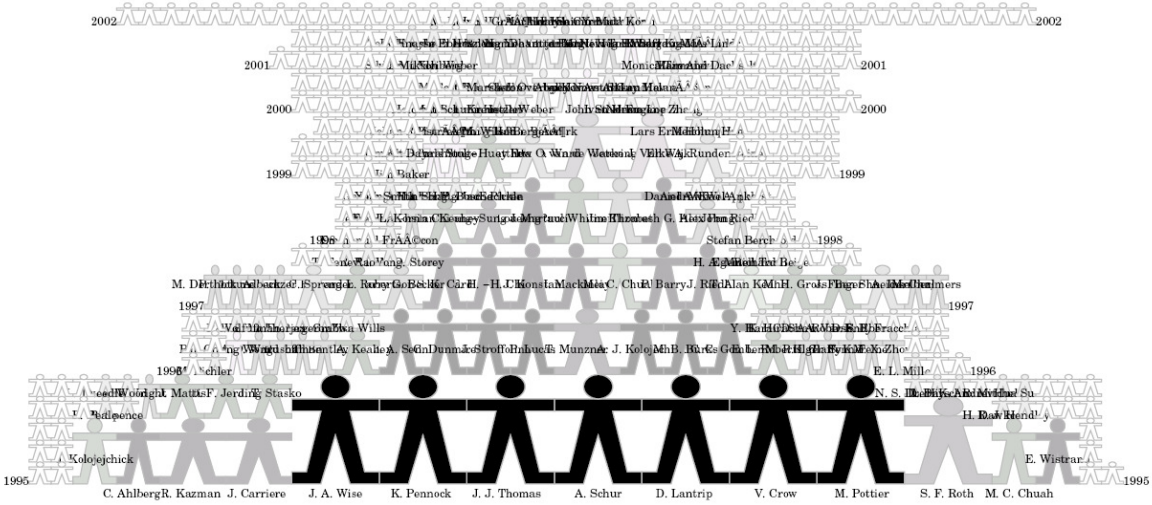


Figure 5.1. Timeline Visualization Example [34]

Interactive Concept Map:

CiteWiz provides an interactive concept map aimed at showing essential research topics and co-authorship. This graphical representation is based on a force-directed graph layout scheme [32] in which a concept is mapped to a node and relationships among concepts are represented as undirected edges. Each node is assigned a weight visually encoded by the node's size, and all edges are modeled as linear springs. A very simple spring simulation is employed to achieve an animation. Several navigation functionalities are also supported.

Figure 5.2 illustrates an example where a concept is a keyword using its total number of frequencies as weight. Two keywords are connected at an edge if they have ever been contained in a research article. This example shows keywords in InfoVis 2004 contest dataset and reflects associated hot topics and themes. In addition, CiteWiz also provides an example interactive concept map in terms of co-authorship; more details can be found in [34].

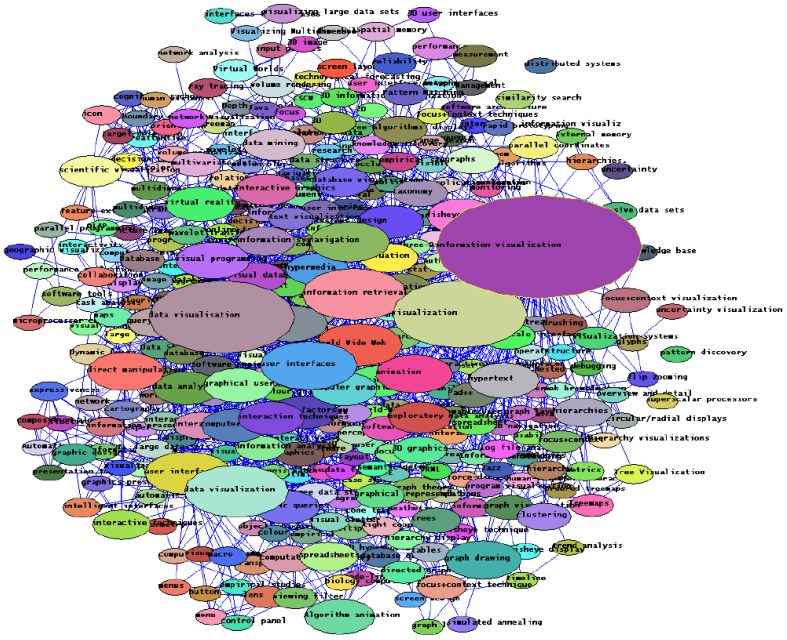


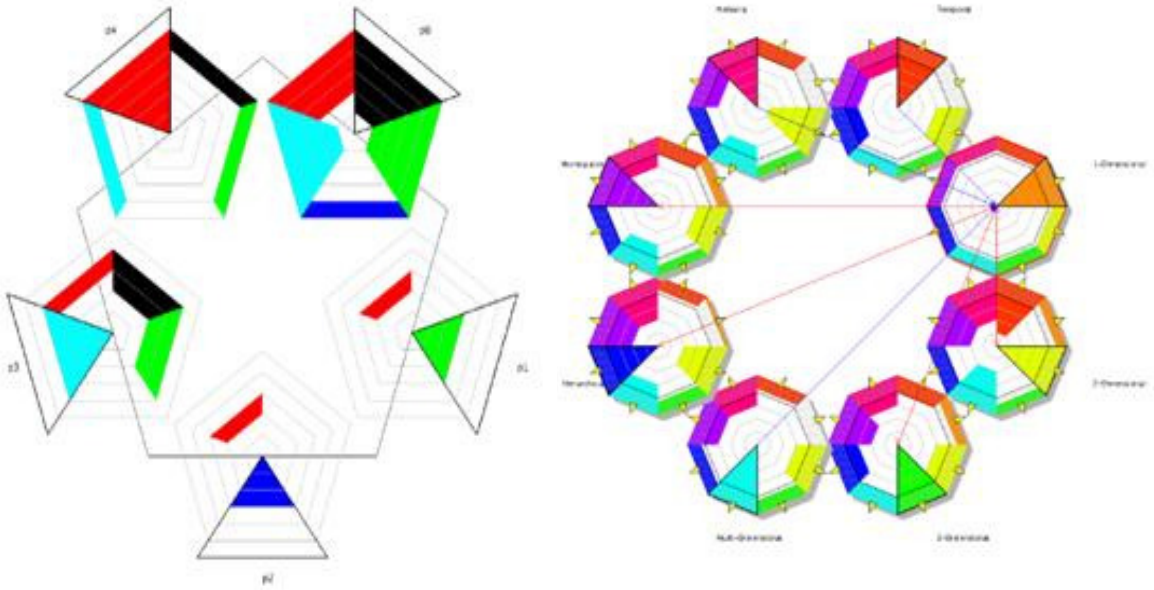
Figure 5.2. Concept Map for the Keywords in the InfoVis Dataset [34].

Influence Visualization

Besides the timeline visualization and the interactive concept map, which mainly provide an overview of a citation database, CiteWiz offers influence visualization which shows citations among a user-defined set of articles. Influence visualization is built based on the idea of Growing Polygons [36], which is usually used to show causality relationships.

Growing Polygons uses *n*-sided 2D so-called process polygons and their associated color to represent the message interaction among *n* active processes. Each process polygon is located along an edge of one big *n*-sided polygon, and a particular triangular sector is applied to illustrate

its position. The size of each process polygon grows over the time; the sectors are filled when a message is received from other processes. The layout in Figure 5.3(a) illustrates a system consisting of five processes. This system ends at a time when process P_o has all its sectors filled, depicting it has received messages from all other processes, whereas process P_1 is only influenced by P_4.



(a) Example Growing Polygons influence visualization with 5 actors/processes [34]

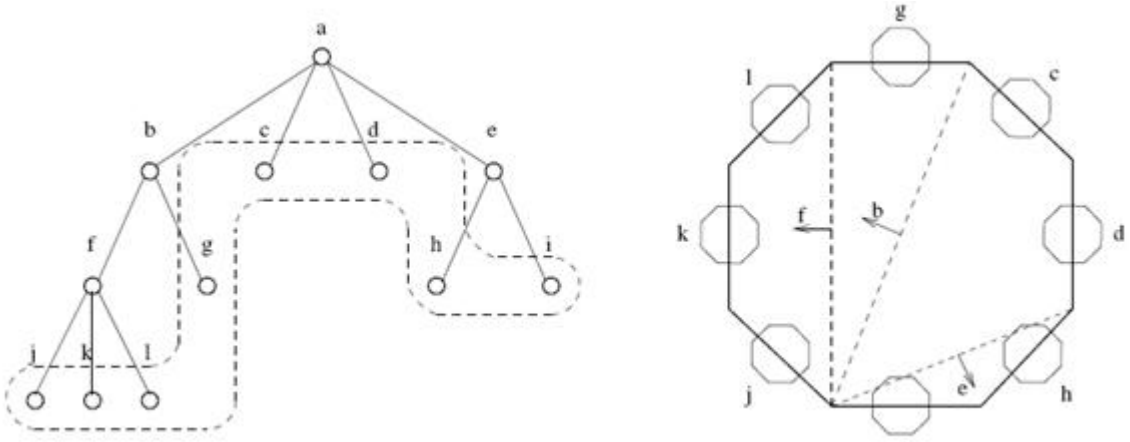
(b) Papers from the Infovis 2004 contest grouped into user defined hierarchical subset representing research areas and visualized using influence visualization [34]

Figure 5.3. Example Growing Polygons

The Growing Polygons technique can be suitably adapted to handle with citation network analysis. Each article is mapped to a process, and a citation is mapped to a message sent from the cited article to the citing article. A citation network therefore mimics message processers' interaction in a system.

The influence visualization in Figure 5.3 (b) visualizes citations grouped into several research fields. The little yellow arrow indicates having a citation to another node, so every field has a

citation to all other fields. When a node is selected (the "1 Dimension" is selected in Figure 5.3 (b)), blue and red edges are drawn indicating cited and citing articles, respectively.



(a) Simple article hierarchy [35].

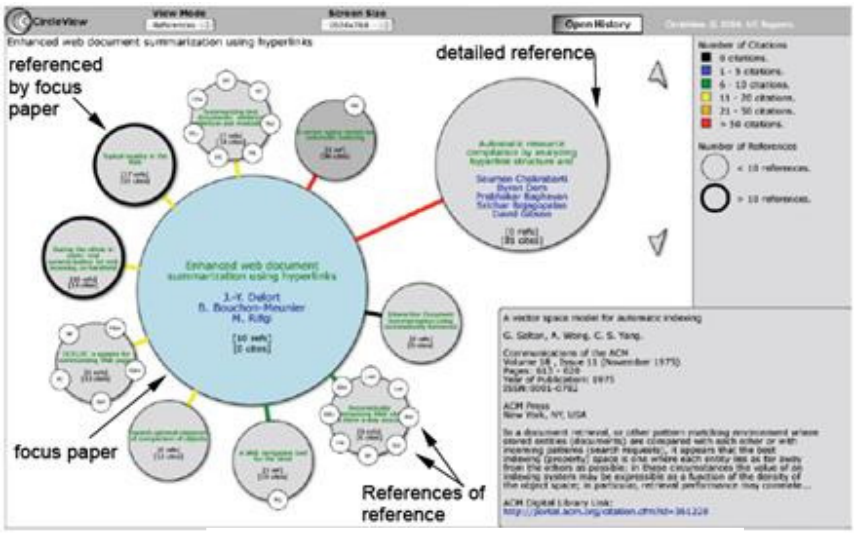
(b) An expanded Growing Polygons diagram, The dashed region in the hierarchy shows the level of expansion, and the dashed lines (chords) in the GP diagram show parent-child relationships [35].

Figure 5.4. Simple Article Hierarchy and Its Growing Polygons Diagram.

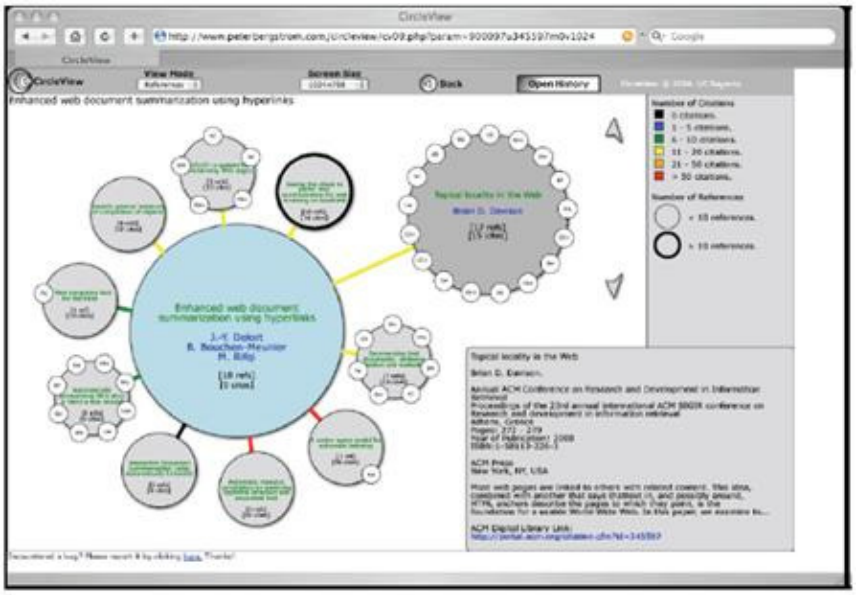
Moreover, influence visualization employs enclosure to indicate the parent-children relationship in a hierarchy. For a set of nodes belonging to the same parent, a chord linking the polygons of the first and last child is drawn. The influence visualization to the right of Figure 5.4 shows such an example. For example, nodes *j*, *k* and *l* belong to the same parent *f*, then a dashed line *f* is drawn to show these nodes are grouped together.

5.2.1.2 CircleView

CircleView [15] is an infovis tool that helps users navigate across a citation network. The paper that a user is currently viewing is called the focus paper. The focus paper is always mapped to a big circle surrounding smaller circles, which denote the second level of citation.



(a) Circle View User Interface [15]



(b) Circle View when the reference circle denoting paper "Summarizing text documents: sentence selection and evaluation metrics" is slected as the detail reference [15].

Figure 5.5. Circle View [15]

Figure 5.5 (a) shows an example CircleView layout using sample data extracted from the ACM Digital Library. The biggest circle in the middle is the focus paper entitled "Enhanced web document summarization using hyperlinks." Surrounding circles are called reference circles denoting those cited by the focus paper. Smaller circles on the reference circles are those circles'

citations. Instead of showing every individual citation, a reference that itself has more than ten citations is drawn in bold line. On the other hand, a reference circle draws its own citation if it has fewer than ten citations. The number of citations between the focus paper and a reference circle is divided into several levels and is represented by color coding. The larger circle in the top right is the detailed reference. Users click a reference circle to turn it to the detailed reference. As shown in Figure 5.5 (b), when a bold-line circle becomes the detailed reference, all of its citations are shown. The text-based panel on the right bottom shows metadata when users move the cursor over a paper. In addition, CircleView provides functionalities such as recording history, serialized navigation and hyperlinks.

5.2.1.3 Journal Citation Report (JCR) Citation Patterns Visualization

Figure 5.6 illustrates a project "JCR Citation Patterns" [145] collaborated on by the Eigenfactor Project [138] and Moritz [148]. This visualization provides an overview of a citation network that is organized into a four-level hierarchy: "Whole research field – Research group – Research field – Individual journals," and only the last two levels are visible.

Color coding is used to distinguish the four research groups. The segments in the outer ring denote research fields which are subdivided into inter segments representing individual journals. The inter segments are scaled by EigenfactorTM Score [139]. Figure 5.6 illustrates the initial view where the top one-thousand citation edges are drawn. Edges are routed along the hierarchy structure by a hierarchy bundling technique [56] that generates bundles between journals. The thickness and translucency of a line indicates the connection strength. Clicking a single journal or a field triggers highlighting all citation flows coming in or out of the selection.

This visualization uses a subset of JCR citation data 1997-2005. More details about the data aggregation and selection can be found in the project description [145].

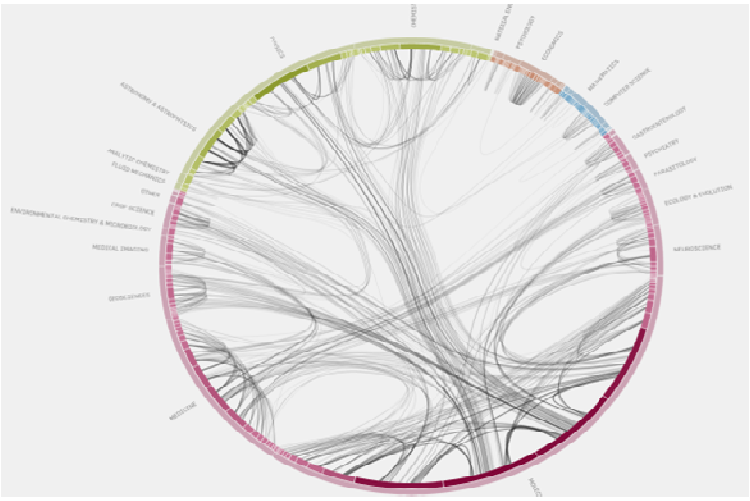


Figure 5.6. JCR Citation Patterns [145].

5.2.2 Journal Citation Reports

The network of journals plays a paramount role in the scientific and technical community and that is why JCR focuses on journal-journal level citations. JCR is probably the most widely used journal-journal citation database from ISI owned by the company Thomson Reuters. Working on the basis of the Science Citation Index in 1961, Price [100] constructed a program mapping the sciences in terms of a collection of structured journal-journal citations as follows:

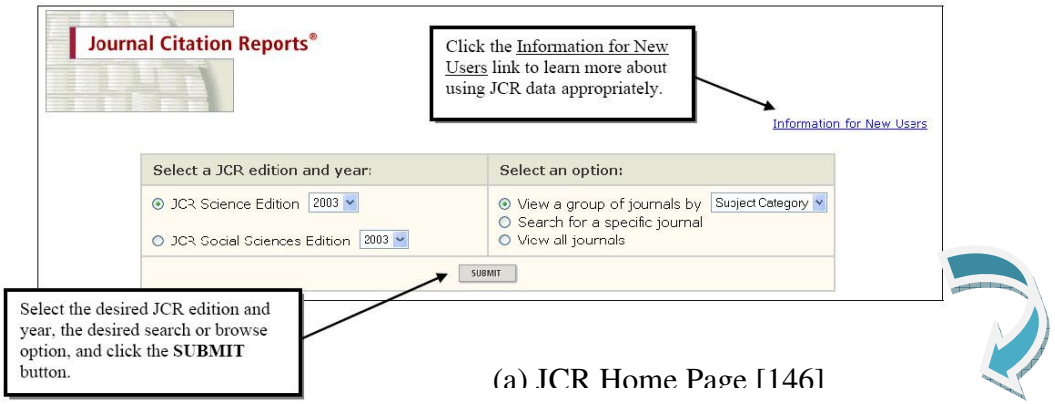
> *The total research front of science has never, however, been a single row of knitting. It is, instead, divided by dropped stitches into quite small segments and strips. From a study of the citations of journals by journals. I come to the conclusion that most of these strips correspond to the work of, at most a few hundred men at any one time. Such strips represent objectively defined subjects whose description may vary materially from year to year but which remain otherwise an intellectual whole. If one would work out the nature of such strips, it might lead to a method for delineating the topography of current scientific literature….*

Journal citations provide the most readily available data for a test of such methods. [100]

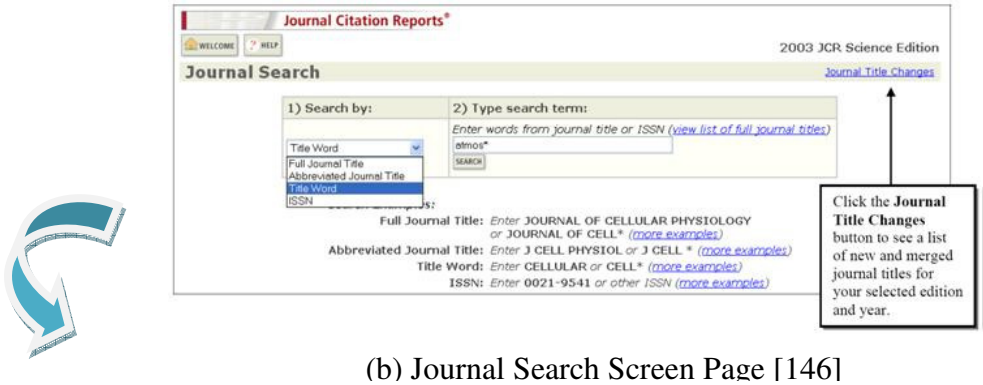
Based on accumulated citation data, JCR provides a systematic and objective way of measuring journals with quantitative assessment. In addition, citations are used to understand relationships between journals, or even on a higher level, such as research field level. JCR's coverage is international and multidisciplinary. So far, JCR has already accumulated citation data of 8,000 of the world's most highly cited journals from 3,300 publishers in about 227 disciplines, from 66 countries. JCR contains two editions published both annually. The science edition covers approximately 6,000 journals in about 170 disciplines, and journals and social science edition covers more than 1,900 journals in more than 50 disciplines.

Although JCR is one of the largest and most comprehensive citation databases, there are some arguments about its usage. The most famous one is that JCR leaves out some important foreign language journals, particularly those which do not use the Roman alphabet. This is true because JCR rarely embodies journals with problems of translation. Logically, this fact ought to negatively affect the usage for a researcher in a country speaking a foreign language, such as Japan or Russia. However, surprisingly, evidence shows that Russian researchers reported that this database satisfies their needs. Moreover, Russian scientists seem to use citation analysis to study science policy more than other scientists do [43].

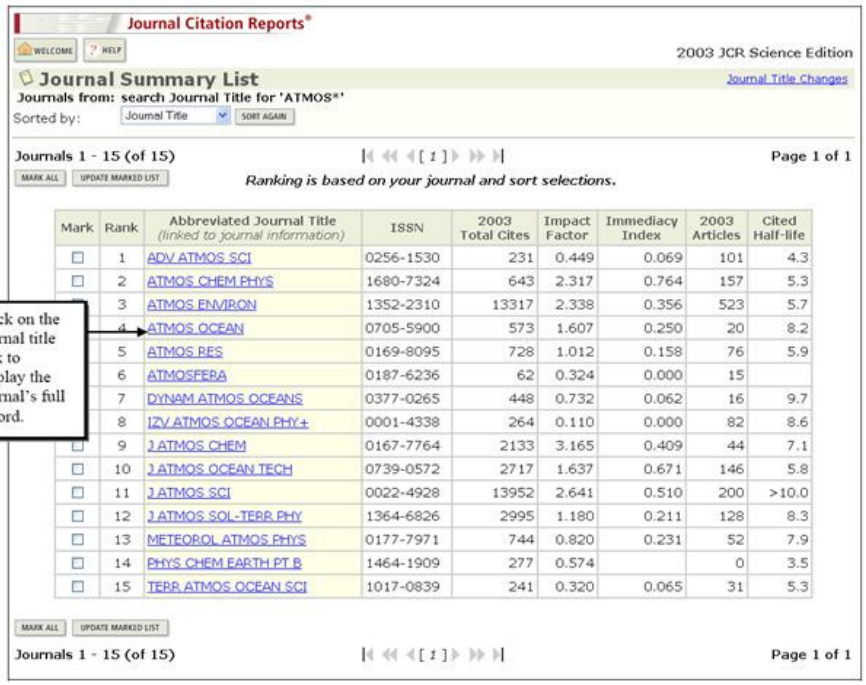
JCR on the Web provides an online interface for users to access the citation database (only to authorized users or institutions). Figure 5.7 shows some example screen shots. Figure 5.7 (a) is the JCR home page where users are allowed to choose an option of viewing journals of a specific edition in a year. Three options are given: "view a group of journal by a selected category, i.e., a research field," "search for a specific journal" and "view all journals."



(a) JCR Home Page [146]



(b) Journal Search Screen Page [146]



Mark	Rank	Abbreviated Journal Title (linked to journal information)	ISSN	2003 Total Cites	Impact Factor	Immediacy Index	2003 Articles	Cited Half-life
☐	1	ADV ATMOS SCI	0256-1530	231	0.449	0.069	101	4.3
☐	2	ATMOS CHEM PHYS	1680-7324	643	2.317	0.764	157	5.3
☐	3	ATMOS ENVIRON	1352-2310	13317	2.338	0.356	523	5.7
☐	4	ATMOS OCEAN	0705-5900	573	1.607	0.250	20	8.2
☐	5	ATMOS RES	0169-8095	728	1.012	0.158	76	5.9
☐	6	ATMOSFERA	0187-6236	62	0.324	0.000	15	
☐	7	DYNAM ATMOS OCEANS	0377-0265	448	0.732	0.062	16	9.7
☐	8	IZV ATMOS OCEAN PHY+	0001-4338	264	0.110	0.000	82	8.6
☐	9	J ATMOS CHEM	0167-7764	2133	3.165	0.409	44	7.1
☐	10	J ATMOS OCEAN TECH	0739-0572	2717	1.637	0.671	146	5.8
☐	11	J ATMOS SCI	0022-4928	13952	2.641	0.510	200	>10.0
☐	12	J ATMOS SOL-TERR PHY	1364-6826	2995	1.180	0.211	128	8.3
☐	13	METEOROL ATMOS PHYS	0177-7971	744	0.820	0.231	52	7.9
☐	14	PHYS CHEM EARTH PT B	1464-1909	277	0.574		0	3.5
☐	15	TERR ATMOS OCEAN SCI	1017-0839	241	0.320	0.065	31	5.3

(c) Journal Summary List [146]

Figure 5.7. JCR on the Web

Figure 5.7 (b) illustrates the example when the option "search for a specific journal" is selected. Users can search a journal by given criterion such as full journal title, abbreviated journal, ISSN and so on. Figure 5.7 (c) shows a list of journals whose titles contain "atmos." Users can sort these resulting journals in different ways and check more details about a journal by clicking on it.

JCR provides several statistical parameters extracted from citation data. All of them can measure the importance of a journal on a particular perspective. Here we only introduce some of them; please refer to [146] for more specification of other parameters.

- Total Cites.

  This is the most basic and also the most important parameter for assessing a journal's importance. Most of the other parameters are derived from Total Cites. Total Cites of a journal is calculated by simply accounting how many times the journal has been cited.

- Impact Factor.

  A journal's impact factor indicates the frequency of average articles in the journal in a year. It is computed by dividing the number of current citations to items which have been published in the last two years by the total number of articles published in the last two years.

- Immediacy Index:

  Immediacy Index of a journal indicates how quickly the average article in a journal is cited. It is computed by dividing the total number of citations to articles which were published last year by the number of articles published in that year.

5.2.3 Computer Science Data in JCR

5.2.3.1 Computer Science Structure in JCR

JCR covers journals in multiple disciplines. As illustrated in Figure 5.8, the computer science data in JCR are organized into a three-level hierarchy. The first level is the entire computer science. The second level consists of seven computer science research areas, which are future divided into individual journals in the third level.

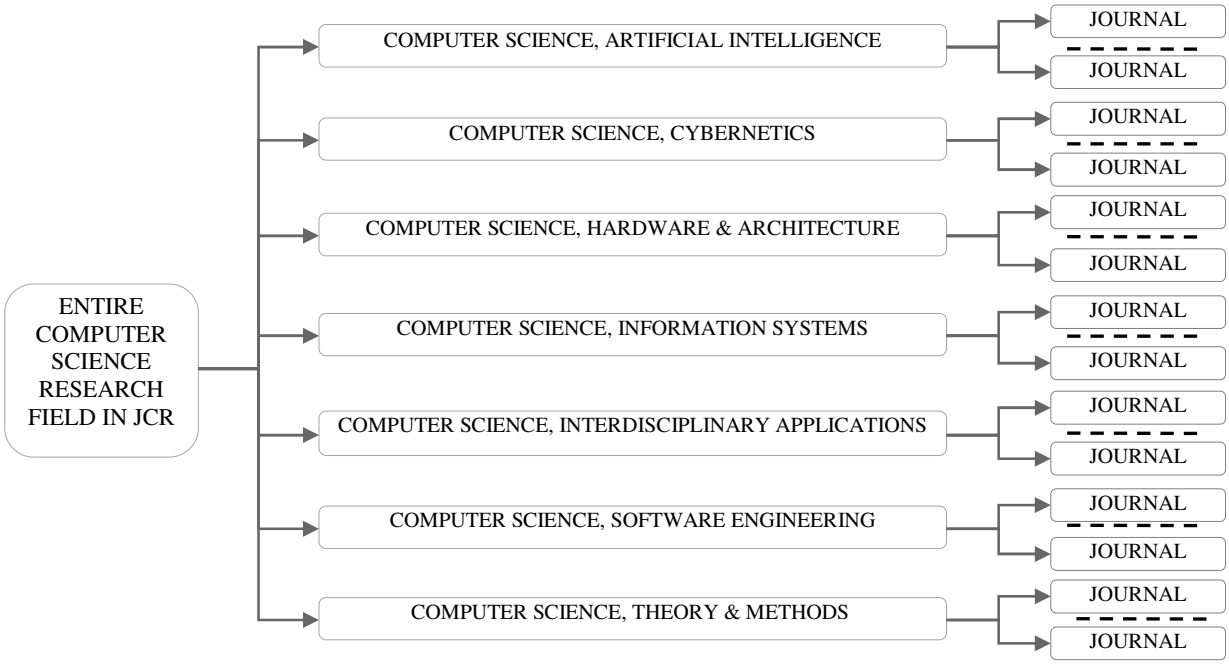
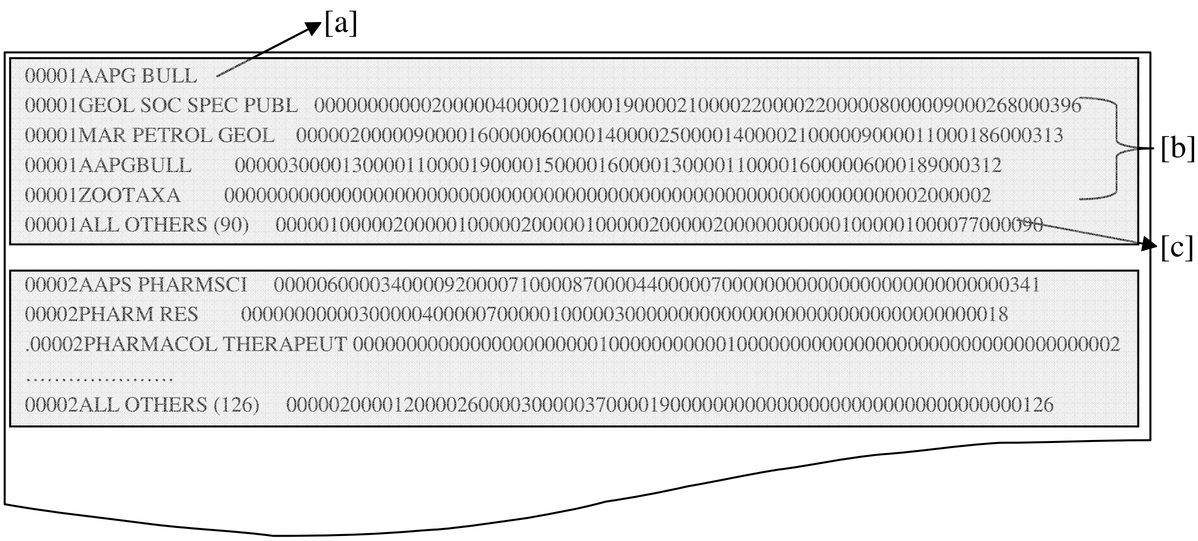


Figure 5.8. Computer Science Structure in JCR

5.2.3.2 Extract Computer Science Data from JCR

As per previous discussion, the research in this dissertation focuses on visualizing the computer science areas' popularity and relationships. First, an indicator parameter should be selected to denote the popularity. While each provided parameter can show a journal's popularity from a certain point of view, we select the prime parameter, Total Cites, as the indicator.

The next step is to extract the computer science citation data from JCR data. Thomson Reuter provides two files, "CITED.dat" containing detail citation data, and "JCR_sci.mdb," which shows index and parameters of journals.



[a]: Summary [b] Subsequence Line [c]: All Others

Figure 5.9. Sample Data in CITED.dat File on the Year of 2004

Title20	Title ID	Title	ISSN	Country	Language	PubID	Categ	Total Cites	Issues	Impact	Immediacy
AAPG BULL	1	AAPG BULLETIN	0149-1423	UNITED STATES	ENGLISH	BH501	ID IP LE	004643	000075	01.140	00.293
AAPS PHARMSCI	2	AAPS PHARMSCI	1522-1059	UNITED STATES	ENGLISH	BH515	TU	000341	000010	01.938	00.600
AATCC REV	3	AATCC REVIEW	1532-8813	UNITED STATES	ENGLISH	BL287	DW II QJ	000139	000082	00.434	00.024
ABDOM IMAGING	4	ABDOMINAL IMAGING	0942-8925	UNITED STATES	ENGLISH	YK501	KI VY	001246	000114	00.884	00.088
ABH MATH SEM HAMBURG	5	ABHANDLUNGEN AUS DEM MATHEMATISCHEN SEMINAR DER UNIVERSITAT HAMBURG	0025-5858	GERMANY	GERMAN	ZX888	PQ	000265	000020	00.146	00.000

Figure 5.10. Example Part of JCR_sci.mdb file in the Year of 2004

CITED.dat file consists of a collection of grouped data. Figure 5.9 shows two grouped data (indicating by two gray boxes) in the CITED.dat file in 2004. A grouped data contains all

citation information of one journal and is composed of lines of data. There are three types of lines in a grouped data: summary line, subsequence line and all others line. The first five characters of a line form the identifier of the cited journal. This identifier can be found in the Title ID file in JCR_sci.mdb file in the respective year. For instance, as illustrated in Figure 5.10, the journal AAPG BULL's identifier is 1.

Bytes	Value	Description
1-5	00001	journal identifier of the **cited** journal. Found on the JOURNAL_MASTER table in the respective year's SCI JCR.mdb.
6-25	AAPG BULL	Journal Abbreviation of the **cited** journal.
26-31	000022	# of citations to the cited journal in the year 2004
32-37	000066	# of citations to the cited journal in the year 2003
38-43	000130	# of citations to the cited journal in the year 2002
44-49	000157	# of citations to the cited journal in the year 2001
50-55	000148	# of citations to the cited journal in the year 2000
56-61	000154	# of citations to the cited journal in the year 1999
62-67	000167	# of citations to the cited journal in the year 1998
68-73	000159	# of citations to the cited journal in the year 1997
74-79	000113	# of citations to the cited journal in the year 1996
80-85	000120	# of citations to the cited journal in the year 1995
86-91	0003407	# of citations to the cited journal in the "Rest", years prior to 1995
92-97	004643	Total citations over all years to this cited journal

Figure 5.11. Explanation of Summary Line

Bytes	Value	Description
1-5	00001	journal identifier of the **cited** journal. Found on the JOURNAL_MASTER table in the respective year's SCI JCR.mdb.
6-25	GEOL SOC SPEC PUBL	Journal Abbreviation of the **citing** journal.
26-31	000000	# of citations to the cited journal in the year 2004
32-37	000002	# of citations to the cited journal in the year 2003
38-43	000004	# of citations to the cited journal in the year 2002
44-49	000021	# of citations to the cited journal in the year 2001
50-55	000019	# of citations to the cited journal in the year 2000
56-61	000021	# of citations to the cited journal in the year 1999
62-67	000022	# of citations to the cited journal in the year 1998
68-73	000022	# of citations to the cited journal in the year 1997
74-79	000008	# of citations to the cited journal in the year 1996
80-85	000009	# of citations to the cited journal in the year 1995
86-91	000268	# of citations to the cited journal in the "Rest", years prior to 1995
92-97	000396	Total citations over all years to this cited journal

Figure 5.12. Explanation of Subsequence Line

Bytes	Value	Description
1-5	00001	journal identifier of the cited journal. Found on the JOURNAL_MASTER table in the respective year's SCI JCR.mdb.
6-25	ALL OTHERS (90)	All other journals that cite the cited journal.
26-31	000001	# of citations to the cited journal in the year 2004
32-37	000002	# of citations to the cited journal in the year 2003
38-43	000001	# of citations to the cited journal in the year 2002
44-49	000002	# of citations to the cited journal in the year 2001
50-55	000001	# of citations to the cited journal in the year 2000
56-61	000002	# of citations to the cited journal in the year 1999
62-67	000002	# of citations to the cited journal in the year 1998
68-73	000000	# of citations to the cited journal in the year 1997
74-79	000001	# of citations to the cited journal in the year 1996
80-85	000001	# of citations to the cited journal in the year 1995
86-91	000077	# of citations to the cited journal in the "Rest", years prior to 1995
92-97	000090	Total citations over all years to this cited journal

Figure 5.13. Explanation of All Others Line

A summary line is the first line of every grouped data and shows the total number of citations to the cited journal by year. For example, the summary line of the first grouped data in Figure 5.9 is "00001AAPGBUL000022000066000130000157000148000154000167000159000113000120003407004643". This summary line can be interpreted by every six bytes. The detailed interpretation can be found in Figure 5.11.

All citing journals in the subsequence line are highly connected to the cited journal. The "all others" lines aggregates the citation data for all citing journals which are very weakly related to the cited journal. The "all others" line is recognized by the phrase "ALL OTHERS" after the fist five bytes. The number in brackets denotes the total number of citations from all other journals which are excluded in subsequence lines. More detailed explanation can be found in Figure 5.13.

A subsequence line is a line of detailed data for one journal citing the named cited journal. The first 25 bytes is the combination of the cited journal's ID and the citing journal's brief name and therefore defines a corresponding relationship. The rest of a subsequence line can be interpreted by every six bytes. Take the subsequence line "00001GEOL SOC SPEC PUBL

00000000000200000400002100001900002100002200002200000800000900026800039 6" for example; the first 25 bytes define a citation relationship between a cited journal whose ID is 1 and a citing journal whose brief title is GEOL SOC SPEC PUBL. The interpretation of the rest of the subsequence line can be found in Figure 5.12.

Table 5.1. Journals in Computer Science Areas in JCR

Computer Science Sub-fields	Number of Journals
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	94
COMPUTER SCIENCE, CYBERNETICS	13
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	44
COMPUTER SCIENCE, INFORMATION SYSTEMS	78
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	74
COMPUTER SCIENCE, SOFTWARE ENGINEERING	47
COMPUTER SCIENCE, THEORY & METHODS	44

A program is written to automatically extract citations in which both citing and cited journals are computer science journals. The resulting extracted computer science citation data covers 384 computer science journals from 1999-2007. Table 5.1 illustrates how many journals are contained in each computer science research area. Please refer to APPDENDIX A for a detailed description.

5.3 Applying 5DITS to Design SciTrend

5.3.1 Define Triggering Problem

The *TP* is defined as "Visualize Popularity Trends of Computer Science Research Areas and Relationships." Two points need to be clarified in advance.

A computer science research area's popularity is its state of receiving tremendous attention and being widely researched. Citation data in JCR offers a natural way of showing popularity at the journal level. The idea is similar to Google's PageRank; we interpret a citation from journal *A* to

journal *B* as a vote, by page *A*, for page *B*, and therefore, the Total Cites essentially represents the popularity of a journal. Although JCR data only show popularity at the journal level, just like a website consisting of many hot web pages must be a hot website, the popularity of a computer science research area can be represented by the sum up of Total Cites of all journals in the research area. Similarly, just as the connection between two websites is built on the individual underlying web pages connections, the relationship between two computer science research areas is also the outer statistical reflection of relationships at journal level.

5.3.2 Decompose the Triggering Problem

This section constructs a logic tree which provides a systematic view of *TP*: "Visualize Popularity Trends of Computer Science Research Areas and Relationships." Modjeska *et al.* [85] proposed a minimum set of functions necessary for effective bibliographic visualization: (1) display of the complete bibliographic entry, (2) filtering by title, author and keyword, (3) display of chronology and influence of articles, i.e., ordered citation links, (4) information views at different levels of detail, (5) multiple simultaneous views and (optionally) synchronized view, (6) visualization of large search result sets. We apply this set as a model to divide *TP* into a set of small sub-problems.

There exist several ways of organizing small sub-problems. Figure 5.14 illustrates an example logic tree which is constructed according to Shneiderman's information seeking mantra [114]. Figure 5.15, however, builds another example logic tree according to the information's degree of detail. The number of the stars behind each sub-problem denotes its weight and also indicates its relative importance to *TP*.

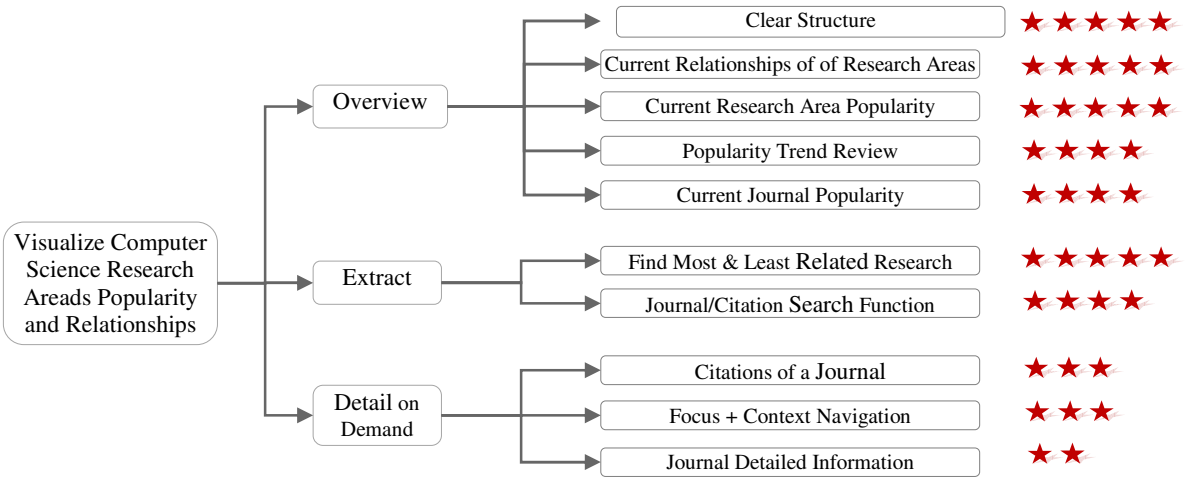


Figure 5.14. An Example Logic Tree Grouped by Tasks

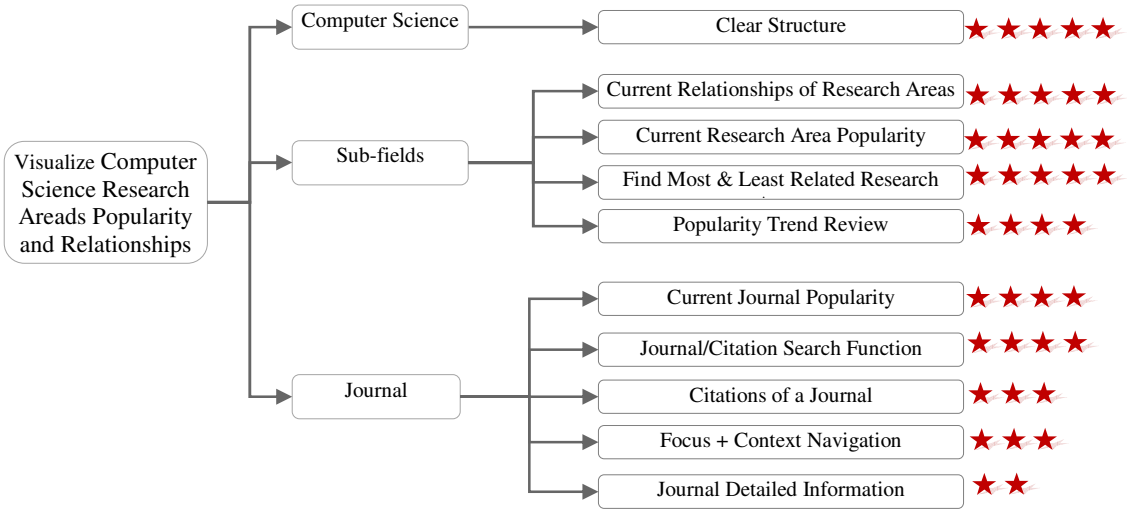


Figure 5.15. An Example Logic Tree Grouped by Degree of Details

Because the primary purpose of our research is to show citation network on research area level together with some less important functionality of view information on journal level, we choose the logic tree in Figure 5.15 which suits our purpose better. Below is the specification of each sub-problem:

- Clear structure.

  The designed infovis technique has to clearly show a hierarchical layout representing the JCR computer science classification system.

- Current Relationships of Research Areas.

  The connections between research areas need to be explicitly visualized. In addition, the visual patent of connections must be able to show the degree of tightness between research areas.

- Current Research Areas Popularity.

  The current popularity of a research area should be encoded as an easily distinguished visual feature so that users can recognize the current popularity difference among different research areas at first glance.

- Find Most/Least Related Research Areas.

  For a research area *A*, *B* is the Most Related Sub-field of *A* if the number of citations between *A* and *B* is no less than the number of citations between *A* and any other research areas. The citations include both ones that journals in *A* cite journals in *B*, and ones in the opposite direction, because we believe both citations indicate that *A* and *B* are somehow related. Research Area *C* is the Least Related Sub-field of *A* if the number of citation between *A* and *C* is no bigger than the number of citations between *A* and any other research areas. There may be more than one Most/Least Related Sub-field for a certain research area. The most and least related research areas of a certain research areas are usually two research areas that users are interested in; therefore, the designed infovis technique should have a function of pointing out these two particular research areas for a particular research area.

- Popularity Trend Review.

  The designed infovis technique should visualize the popularity changes of all research areas over time. In addition, the related popularity of different research areas in a year should be recognized.

- Current Journal Popularity.

  Popularity at the journal level should also be easily distinguished. Believing that showing everything makes visualization showing nothing, the designed visualization technique intends to visually group journals according to their popularities, rather than trying to show tiny popular difference of every pair of journals.

- Journal/Citation Search Function.

  Users should be able to search a journal or a citation by key words. The found results should be highlighted in a reasonable manner.

- Citations of a Journal.

  The designed infovis technique should provide a proper navigation that allows users to explore all citations for a selected journal.

- Focus + Context Navigation.

  A focus plus context technique shows the information of primary interest in full detail while maximally retaining the original context-or overview by compressing all the surrounding information. This "seeing the trees without missing the forest" technique is required mostly under two conditions: first, users need to view both detail information (focus) and overview (context) at the same time. Second, not all information details can be encoded as visual features due to certain reasons, such as the limitation of display space.

- Journal Detail Information.

  Graphical representation is adept at accelerating information comprehension by abstracting details. It is common that small differences among underlying information can hardly be spotted by encoding visual features. However, users demand viewing information in the most

detailed level such as numbers. According to these facts, a text-based panel showing details of currently explored information needs to be embedded in the visualization tool.

5.3.3 Design a Temporary Infovis Technique

This section first seeks for possible *EMs* for each aforementioned sub-problem, followed by a discussion of possible existing infovis techniques. A designed temporary infovis technique is introduced at the last part.

5.3.3.1 Find Existing Solution Method for Each Sub-problem

- Possible *EMs* for sub-problem: "Clear Structure"

  Various techniques have been invented to visualize hierarchical structure and can be generally classified into connection and space-filling approaches. Connection approaches draw a hierarchy as a node-link diagram; space-filling techniques use enclosure instead of links to represent the connectivity. A detail description about these two kinds of approaches is in Section 3.2. Because the node-link diagrams usually have more desirable structural clarity than layouts produced by space-fillings approaches, we choose connection approaches to visualize hierarchy structure in our case.

  Possible *EMs*: {Connection approaches, such as: Classic tree [103] [124], Radial tree [38][53] [54][7], Balloon tree [53][63][84][121]}

- Possible *EMs* for sub-problem: "Current Relationship of Research Areas"

  This sub-problem needs an *EM* which shows citations as links. Next citation links between journals need to be bundled. The hierarchical edge bundles technique [56] models each citation link as a *B*-spline curve so that the implicit adjacency between parent nodes (research areas) can be illustrated by the explicit citation links of child nodes (Journals).

  Possible *EM*: {Hierarchical edge bundles technique [56]}

- Possible *EMs* for sub-problem: "Current Research Area Popularity"

  Current research area popularity should be encoded into the most salient visual feature so that the popularity difference between research areas and the relative importance of a research area to the entire computer science field can be easily recognized. We choose the following distinguished visual features in the *EM* set.

  Possible *EMs* (Encoding visual feature): {Color, Size, Length, Angle, Shape}

- Possible *EMs* for sub-problem: "Find Most & Least Research Area"

  A search bar should be supported to find the most and least related research areas for a particular area and the results should be properly highlighted.

  Possible *EMs*: {Search function & Appropriate highlight function}

- Possible *EMs* for sub-problem: " Popularity Trend Review"

  A review of the popularity trend over time can be shown either by animation or by static depiction. Although animation gains prominence, research shows [106] that static representation accelerates the comprehension of information significantly faster than animation does.

  Possible *EMs*: {A static trend plot method}

- Possible *EMs* for sub-problem : "Current Journal Popularity"

  Current Journal Popularity, measured by a journal's total cites, should be clearly encoded by a visual feature. As per previous discussion, the expected infovis technique only show journal popularity in term of pre-defined levels, rather than try to visualize every small difference of popularity between journals. The possible *EMs* (Encoding visual features) are shown as below:

  Possible *EMs* (Encoding visual feature): {Color, Size, Length, Angle, Shape}

- Possible *EMs* for sub-problem: "Journal/Citation Search Function"

  A search bar is required to search journals or citations by keywords and the results are highlighted.

  Possible *EMs*: {Search function & Highlight function}

- Possible *EMs* for sub-problem : "Citations of a Journal"

  The designed infovis technique needs to provide a way for users exploring citations of a journal. First, an appropriate interaction is needed for users to select a journal, second associating citations are highlighted.

  Possible *EMs*: {Interaction & Highlight Function}

- Possible *EMs* for sub-problem: "Focus + Context Navigation"

  Two of the mostly widely used focus + context navigation techniques, fisheye and hyperbolic tree, are proposed as *EMs*.

  Possible *EMs*: {Fisheye, Hyperbolic tree}

- Possible *EMs* for sub-problem: "Journal Detail Information"

  Although the expected infovis technique can visually show a journal's popularity and citation connections with other journals, a great deal of other information, such as exact number of citations and associated impact factors, is missed. Therefore, an assisting text-based panel should be embedded.

  Possible *EMs*: {A Text-based panel}

5.3.3.2 Possible Exiting Infovis Technique

Instead of designing a new temporary infovis technique by combining *EMs* for each sub-problem, which takes potential risks and uncertainty, we assess the possibility of using existing infovis techniques discussed in previous sections.

CiteWiz, especially the influence visualization, has a high potential for being used as a temporary infovis technique. This influence visualization can be easily adjusted to adapt to our needs. A citation network consisting of *n* journals can be visualized as an influence visualization which is composed of *n* *n*-sided polygons. Each polygon denotes a journal. The degree of fullness of a specific triangular sector in each polygon indicates the popularity of the associated journal. In addition, besides the specific triangular sector, other triangular sectors of a polygon denote which journals have citation connection with the associating journal. Users can view citations of a journal by selecting it; blue and red edges indicate cited and citing journals respectively. A search bar, fisheye, and a text-based panel can be easily added. Using chord links to denote parent-child relationships in computer science classification system works well, especially for the simple system contained in JCR. While most of these advantages are on journal level, they can hardly be used on the research area level. For example, a research area's popularity is encoded in a set of dispersedly allocated triangular sectors, and is very hard to be distinguished. Another example is that relationships of research areas are hardly shown. However, this can be solved by replacing the straight line with curves drawn by hierarchy bundle technique [56]. Moreover, influence visualization is also not perfect in showing the citation network on the journal level. When the number of journals becomes large, the size of polygons dramatically shrinks to a degree where triangular sectors are hardly recognized. Therefore, when there are huge numbers of journals, using polygons tends to be unacceptable.

CircleView provides a great way of navigating a citation network. Users have a focus plus context like layout: the focus paper is located at the center surrounded with two levels of citation related papers serving as context. However, the desirability of the layout exists only when CircleView is applied on showing a partial citation network. The structure will go blurry and the

circles become too small to be labeled when this technique is scaled to the entire citation network. By and large, CircleView is built for navigation by showing a partial citation network rather than for visualizing the entire citation structure and inner connections.

JCR Citation Patterns Visualization shows a clear overview. All journals are radically located along a circle as small segments. Unlike CiteWiz which adds a chord link denoting a parent-child relationship, JCR Citation Patterns Visualization simply uses color coding which is more esthetically pleasing. The connection of two research areas is perfectly represented by bundles. JCR Citation Pattern Visualization uses an angle assigned to a segment (small segments denote journals, big segments denote research areas) to represent popularity. This works well for denoting popularity differences of research areas. However, users can barely notice the popularity difference of journals, especially for most whose popularity is very small. From a previous discussion, we believe JCR Citation Patterns Visualization has an advantage over the other two serving as a temporary infovis technique in our case.

5.3.4 Evaluate the Temporary Infovis Technique

The JCR Citation Patterns Visualization solves most high-weighted sub-problems and is a good start for deriving a more appropriate infovis technique. In this section, we evaluate the suitability of this visualization and calculate its *DoS*.

Figure 5.16 illustrates how well the JCR Citation Patterns Visualization performs on *TP*. The gray rectangles indicate the sub-problems remaining unsolved. The stars and the percent behind the solved sub-problems denote the weight and *dos*, respectively.

Regarding the issue of clear structure, a radial tree layout together with an outer ring is applied to show a hierarchy of four levels: “Entire computer science field – Group – Sub-field – Journal.”

In our scenario, the group level does exist. This fact makes the outer ring unnecessary in the layout and cuts off ten percent from the *dos*.

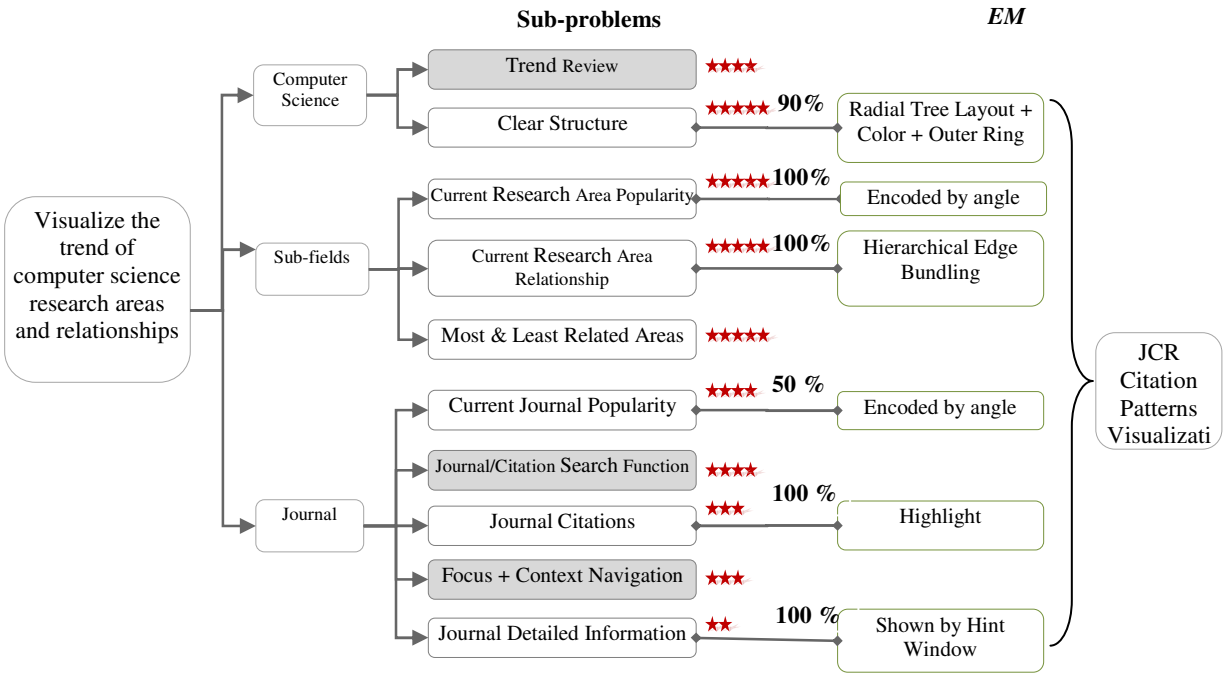


Figure 5.16. Evaluating JCR Citation Pattern Visualization

A research area's current popularity is fairly obviously encoded by the length of an outer ring segment, in essence by the assigned angle. The difference of angles is easily perceived, so the *dos* deserves a full score.

The hierarchy edge bundling technique [56] produces an esthetically pleasing connectivity pattern between research areas and therefore *dos* is full score. The citations between two journals are bundled according to which research area the journals belong to, so the connections between research areas are easily understood. Usually the wider the bundle of citation links between two research areas, the more tightly these two research areas are related; similarly, a narrow bundle of citation links denotes that two research areas are weakly related.

Similar to the current research area popularity, the current journal popularity is also encoded in the angle. However, unlike the big angles which are assigned to research areas, the angles assigned to journals are very possibly quite small and the difference is hardly perceived. Due to the undesirable performance, we believe this angle encodement only can be accepted with a grain of salt and give *dos* half of full score.

When a journal is selected by a single click, the citation links pointing to or from it are clearly highlighted. The *dos* is given full score.

A journal's detailed information is clearly shown in a hint window which is around the cursor when the cursor hovers over a journal. The *dos* is given full score. However, this hint window causes the only conflict with showing current journal popularity. Because the hint window usually overlaps with a partial layout, some displayed journals are out of sight. In our perspective, we consider this conflict of little importance. The Degree of Conflict (*doc)* of this conflict is 10%.

Alluded to above, the total *Dos* of JCR Citation Patterns visualization for P can be computed by Equation 5.1.

$$
\begin{aligned}
Dos &= \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k} \\
&= \frac{(5*0.9+5*1.0+5*1.0+4*0.5+3*1.0+2*1.0)-(2+4)*0.1}{5+5+5+4+5+4+4+3+3+2} \\
&= 49.75\%
\end{aligned}
$$

Equation 5.1. *Dos* of JCR Citation Patterns Visualization.

5.3.5 Improved JCR Citation Pattern Visualization

In this section, we realize improvements based on the evaluation discussed in the previous section.

- Improvement 1:

  Figure 5.17 shows the initial layout, which consists of concentric circles. Each circle represents one year and has seven segments denoting seven computer science research areas. Color coding denotes a research area's type and assigned angle of a segment is scaled to the research area's popularity in that year. Users can click a circle to approach a closer view in that year.

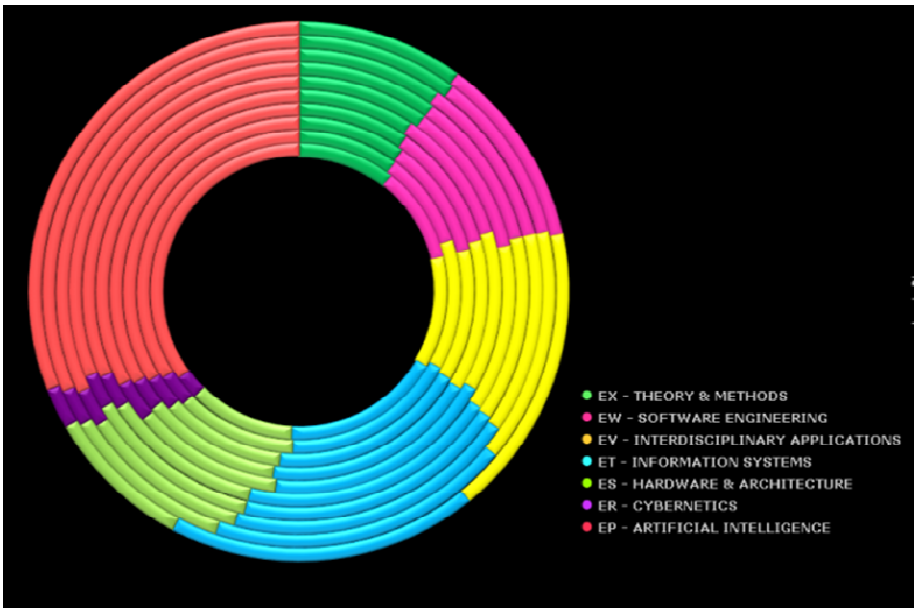


Figure 5.17. Research Areas' Popularity Trend over the year 1999-2008

We avoid using animation and widely accepted trend visualization techniques such as scatter plot for two reasons. First, although animation has gained prominence for showing trend, research [106] shows that it is much less effective than static representation for analysis and trends assimilation. Second, while static trend visualization methods such as scatter plot are neat and widely accepted, users have to make extra cognitive efforts to understand the transition from the non-circular scatter plot to the circular layout. In other words, the disadvantage of conflict between the non-circular overview and the detailed circular layout is too big to be accepted. On

the other hand, even this concentric circle layout is not the best. Although the cognitive gap is very small when users drill down from overview concentric circles to the detailed circular layout, the inherent popularity trend is fuzzy. As per the aforementioned discussion, infovis design is something about tradeoff balancing. Accordingly, we believe the concentric circle layout suits our scenario at this stage. We give *dos* 80% due to the undesirable popularity illustration.

- Improvement 2

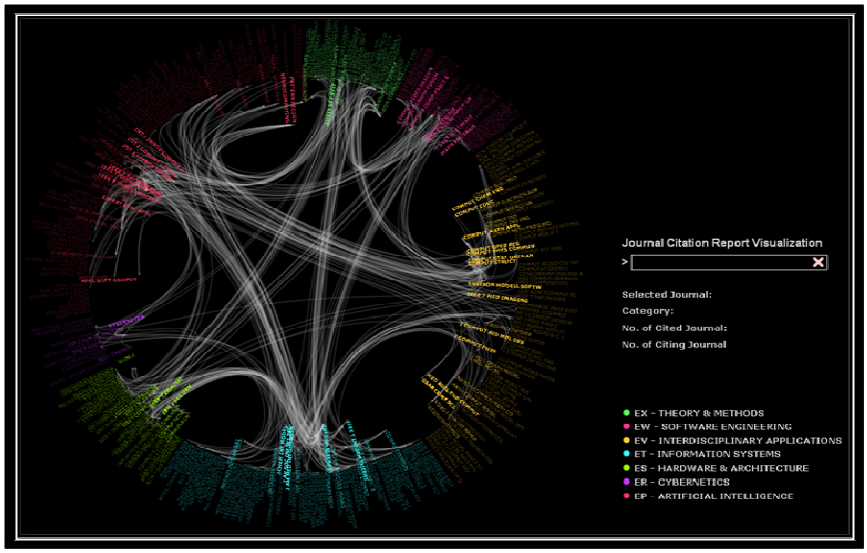


Figure 5.18. Improved JCR Citation Patterns Visualization on 2008 Citation Data

We get rid of using outer rings in the original JCR Citation Patterns layout. The computer science classification system is shown as a radial tree where only nodes at the last level (journals) are visible. Figure 5.18 shows the layout of Improved JCR Citation Patterns Visualization on 2008 citation data. The ring as a whole denotes the entire computer science, and color coding represents the type of research areas. Data of Improved JCR Citation Pattern Visualization covers from 1999-2008. More details are available in Appendix One.

- Improvement 3

The current journal popularity is represented by radius together with transparence. Unlike the original JCR Citation Visualization's attempt of showing very tiny popularity difference

between journals, we classify the journal's popularity into three levels. The journals with higher popularity approach nearer to the center with a clearer look. The journals with least popularity are located far away from the center with fade-out effect. Using radius plus transparence impresses the users by clearly distinguishing the journals' popularity level by level.

- Improvement 4

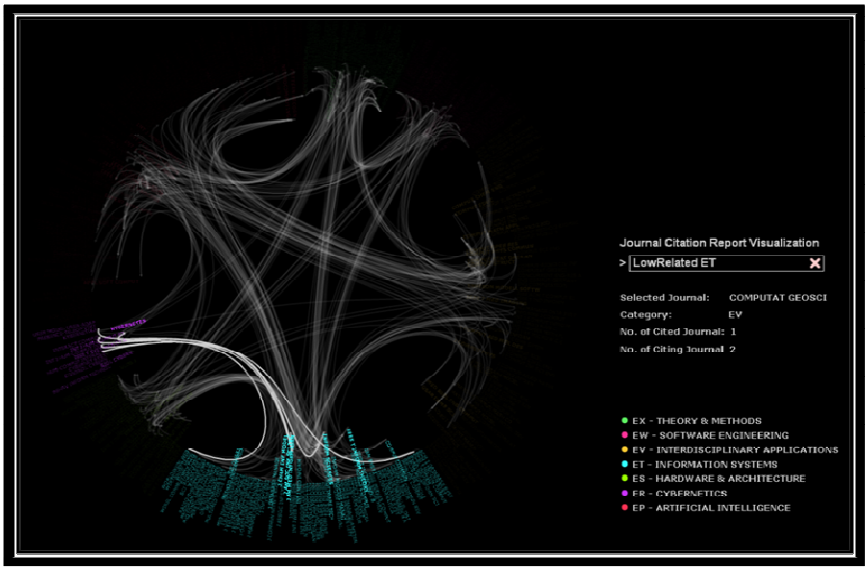


Figure 5.19. The Least Related Research Area Search Example

We apply search and highlight functions to indicate the most and least related research areas of a particular research area. A search bar is implemented on the right hand side of the layout. The simple syntax is "MostRelated/LeasetRelated + research area name." One feature that has to be pointed out is that the research area name is indexed as two letters in the JCR data base, for example, the research area of "Theory & Methods" is indexed as "*EX*". The resulting research area and corresponding citation links are highlighted. Figure 5.19 illustrates an example of finding the least related research area of Information System (*ET*).

- Improvement 5

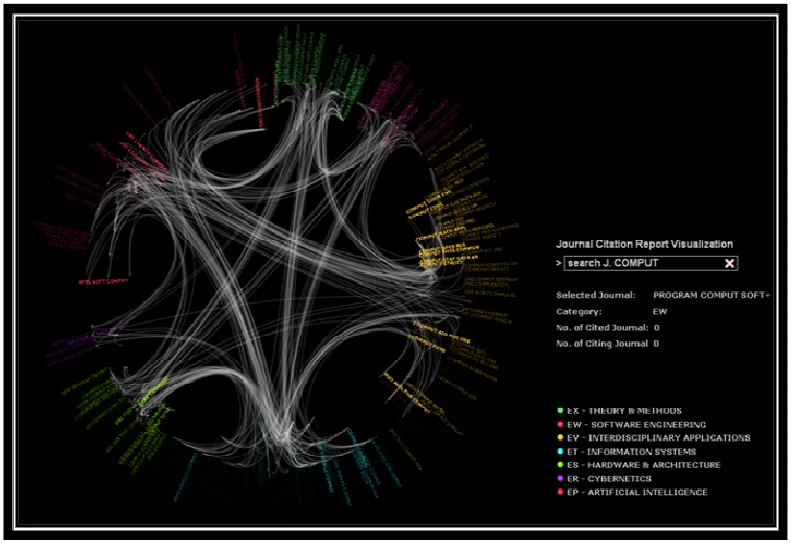


Figure 5.20. Journal Search Example

Besides finding the most and least related research areas of a particular research area, users also can use the search bar to locate specific journals or citations using key words. The syntax of finding specific journals is "search J. + key words." Figure 5.20 illustrates a case of searching the journals whose titles contain "COMPUT." The resulting journals are highlighted and others completely fade out.

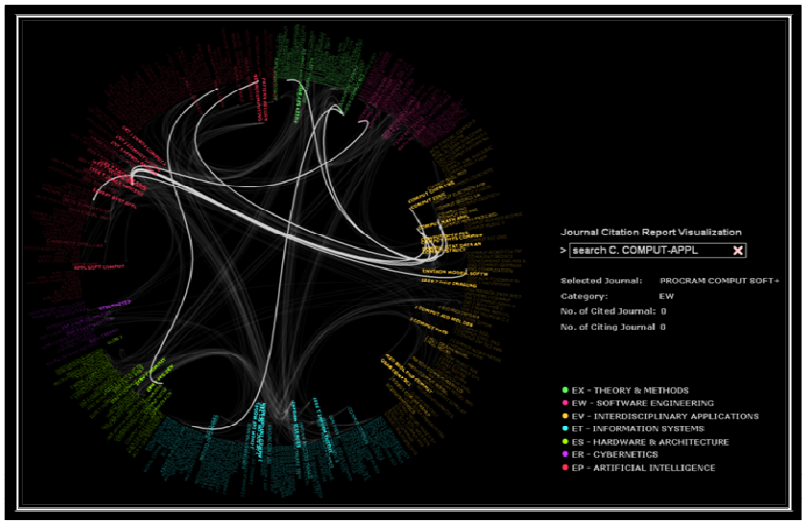


Figure 5.21. Citations Search Example

The syntax of searching citations is defined as "search C. keyword1-keyword 2" where keyword1 and keyword2 are strings that are expected to be contained in the titles of citing and cited journals, respectively. Figure 5.21 represents an example in which all citations that the titles of citing journals contain "COMPUT" and titles of cited journal containing "APPL" are highlighted.

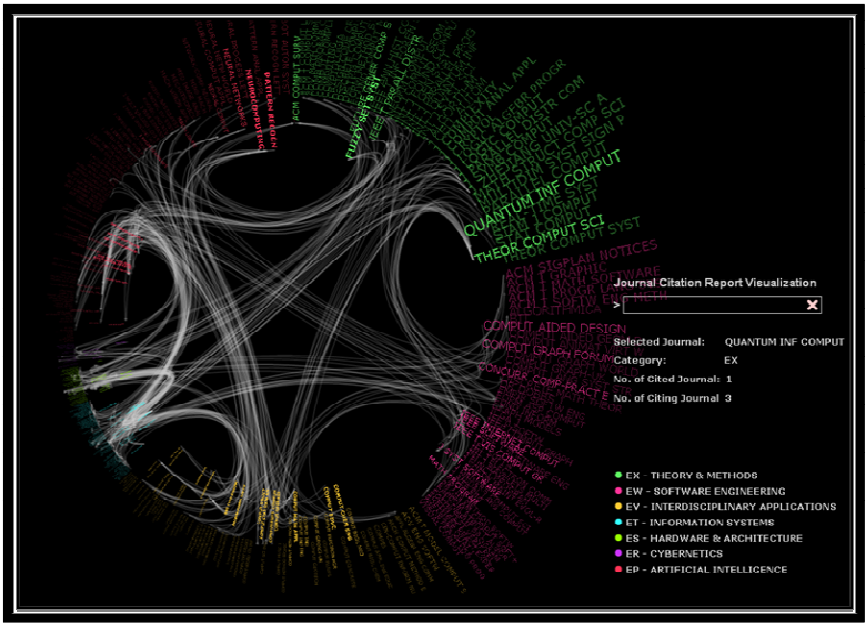


Figure 5.22. Fisheye Layout Effect Example

- Improvement 6

  Fisheye is implemented to enable focus plus context browsing. The fisheye layout is triggered by double clicking a journal. The double clicked journal is the focus. Journals around the focus are assigned bigger space and larger font by compressing the space for journals far away from the focus. Figure 5.22 illustrates an example where the journal "QUANTUM INF COMPUT" is the focus. While fisheye enhances the comprehension of focus with context, it obviously ruins the radial layout encoding research areas' popularity. The *doc* of conflict between fisheye and radial layout is 30%.

- Improvement 7

  A text-based panel under the search bar is supported to show detailed information of journals. It hints a journal's title, research area, number of citing journals and number of citing journals when the cursor hovers over that journal.

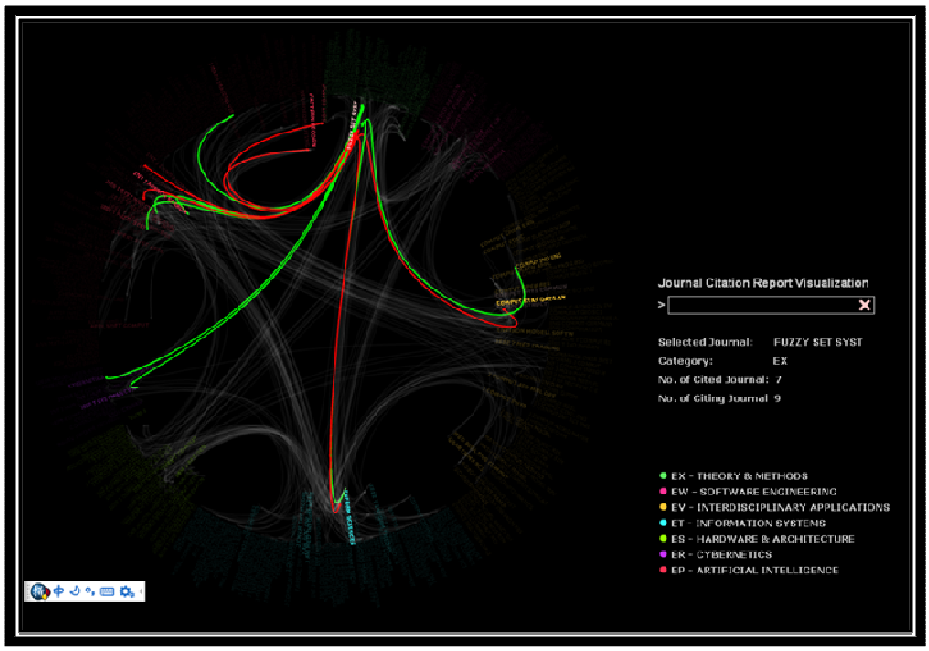


Figure 5.23. An Example Exploration of All Citations for a Journal

Besides these improvements, some notices have to be made. Users can explore the citations of a particular journal by a single click on it. As illustrated in Figure 5.23, the red links represent the cited links and green links represent citing links.

5.3.6 Evaluation of Improved JCR Citation Pattern Report

Figure 5.24 describes how the improved JCR citation pattern visualization solves *TP*: "Visualizing computer science research areas popularity and relationships."

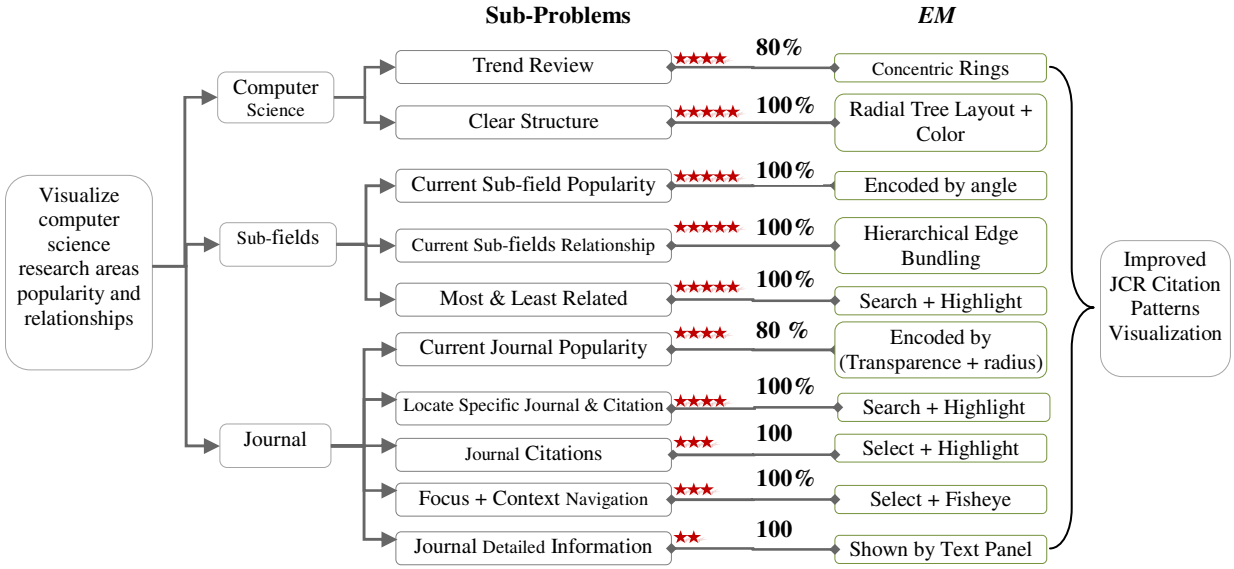


Figure 5.24. Evaluating Improved JCR Citation Pattern Visualization

According to Equation 5.2, the *Dos* of improved JCR citation patterns visualization can be computed as follows:

$$Dos = \frac{\sum_{k=1}^{n}(w_k * dos_k) - \sum(w_i + w_j) * doc_{ij}}{\sum_{k=1}^{n} w_k}$$

$$= \frac{(5*1.0+5*1.0+5*1.0+4*0.8+5*1.0+4*1.0+4*1.0+3*1.0+3*1.0+2*1.0)-(5+3)*0.3}{5+5+5+4+5+4+4+3+3+2}$$

$$= 90.02\%$$

Equation 5.2. *Dos* of Improved JCR Citation Patterns Visualization

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This chapter concludes the dissertation, lists research contributions and discusses some future work.

6.1 Summary of Research

The research presented in this dissertation is one of the first efforts towards putting strategic thinking that is usually encountered in business consulting into the process of designing an infovis technique. Unlike an infovis technique in the evaluation stage in which empirical study can be conducted and real users' evaluation feedbacks can be utilized, an infovis technique in the design is only documented and is unimplemented. Therefore, most current user-based infovis evaluation and improvement methods cannot be applied in designing an infovis technique. To achieve the purpose, the similarities between consulting a business case and designing an infovis technique are discussed. Discovering that both the business consulting and infovis design are triggered by a particular problem, this dissertation constructs a problem-based guideline for designing an infovis technique. First, a classic business problem-solving tool, Mckinsey's 7 steps, is introduced. Second, the strategic thoughts in Mckinsey's 7 steps are compiled into a guideline 5DITS according to the needs of infovis design. 5DITS provides a standard guideline for designing an infovis technique for a triggering problem. It usually shows its power when a new infovis technique is designed from scratch, or user evaluation cannot be conducted to

provide improvement suggestions. The major contribution of 5DITS is that, it clearly points out that designing an infovis technique is not only associated with visual metaphors design, but also with systematic analysis, priority analysis, compromise, efficiency and so on, i.e., strategic thoughts are needed.

Three infovis techniques are designed following 5DITS. Radial Edgeless Tree (RELT) is designed for visualizing hierarchical information on mobile interfaces. The elegance of this technique includes: recursive division of the display area, space-filling, maximum screen space usage, and clarity of the hierarchical structure. RELT has its flexibility greatly extended so that users can customize the root location and stylize the layout. The RELT drawing algorithm is adaptable and customizable for different applications. We demonstrate the algorithm's application for stock market visualization. Moreover, we present an analytical and empirical comparison of our implementation on an emulator with a currently used cell phone interface in terms of their performance in navigation.

InfoShape presents new approaches to visualizing multi-dimensional information as a three-dimensional sphere according to a pre-defined set of criteria. In one aspect of the invention, if a multi-dimensional information set satisfies a given set of criteria it will be represented as a perfect sphere. Otherwise, the distortions on the sphere indicate defective contents whose information does not satisfy some of the criteria. In another aspect of the invention, many sets of multi-dimensional information can be compared by simply comparing the corresponding spheres.

SciTrend, which is derived from journal citation patterns [145], visualizes the popularity trends of computer science research areas and relatonships. The underlying data refer to a journal citation network which covers research articles in these research areas. Users can get a quick

glimpse of a computer research area's popularity trend over years, understand the relationships among these areas, explore connectivity of a particular research article, search an article or a connection by keyword, and so on.

6.2 Contributions

This dissertation contributes the guideline 5DITS and three infovis techniques that are designed following 5DITS. To our best knowledge, 5DITS is the first guideline comprising strategic thinking. In summary, 5DITS guideline's contributions to knowledge include:

- A standard infovis technique design pipeline, which designers can follow to create an infovis technique for a particular triggering problem. 5DITS has proved itself as a practical and problem-centered tool designing and evaluating an infovis technique before implementing it.
- 5DITS does not treat a triggering problem as a simple indivisible unit, but regards a triggering problem as an organic unit consisting of smaller problems twisted with each other. Each smaller problem shows an aspect of the triggering problem. In 5DITS, a triggering problem is analyzed and then is described using a logic tree that provides a systematic view of the triggering problem.
- 5DITS clearly shows that strategic thinking is necessary in the process of infovis technique design. The major evidence is that a logic tree very likely contains pairs of smaller conflicting problems, which are possibly solved well one by one, but cannot be solved perfectly simultaneously. The designed infovis technique needs to strategically solve these smaller problems according to priority, rather than evenly taking every smaller problem.
- The generated logic tree in 5DITS systematically shows all aspects of the triggering problems. It provides designers a chance to maximally use the existing infovis techniques so that the goal of high efficiency can be achieved.

- 5DITS offers a quantitative assessment of an infovis technique. A number is calculated to denote the degree of satisfaction of a designed infovis technique for a triggering problem, i.e., how well an infovis technique solves the triggering problem.

Besides the above contributions of 5DITS, this dissertation also contributes three infovis techniques designed by 5DITS for three different triggering problems. The contributions of these three infovis techniques are as follows:

- This dissertation presents RELT for visualizing hierarchical information on small screens. Major advantages of the RELT approach include: elegant recursive division of the display area, space-filling, maximum usage of screen estate, and clarity of the hierarchical structure. It offers the flexibility such that users can customize the hierarchy's root location and stylize the layout. The RELT drawing algorithm is adaptable and customizable for different application domains.
- This dissertation presents Record InfoShape (RInfoShape) and Dimension InfoShape (DInfoShape), two multi-dimensional information visualization techniques that aim at visualizing multi-dimensional information as a 3D sphere whose appearance denotes how much the provided information satisfies a pre-defined set of criteria. By comparing the shapes of different multiple information sets, overall content similarities and differences can be quickly captured.
- This dissertation dessigns SciTrend aiming at visualizing the popularity trends of computer science research areas and relationships. SciTrend is derived from JCR Citation Patterns [145] and provides various functions. Users can get a quick glimpse of a computer research area's trend over years, understand the relationships among these areas, explore connectivity of a particular research article, search an article or a connection by keyword, and so on.

6.3 Future Work

Regarding the 5DITS, future work includes precisely defining triggering problem scope, developing a better ranking system, trying maximally reducing subjectivity, and using new algorithms to calculate *Dos*.

For example, as mentioned in the dissertation, the triggering problem definition is the key step of 5DITS. A smaller scope will lead to an infovis technique that cannot satisfy all real requirements; however, an oversized scope inevitably generates an infovis technique with unnecessary functions. Future 5DITS should provide principles required to be followed to define the triggering problem. Current 5DITS simply ranks decomposed smaller problems according to their importance to the triggering problem. This ranking method is simple but needs further enhancement. Parameters, such as *dos* and the importance of a decomposed smaller problem are assigned by a designer's personal impression. Although subjectivity cannot be 100 percent avoided, it can be maximumly reduced. Considering the *Dos* which illustrates how well a infovis technique solves its triggering problem, new algorithms accepting more parameters should be developed to computer *Dos*.

For the three designed infovis techniques, RELT has three major possible future improvements: first, the current RELT works well on a balanced tree; but its esthetically pleasing layout turns undesirable when it is applied on an unbalanced tree; second, new algorithms need to be designed to put textual labels within nodes; third, more visual features may be used to denote more or emphasize particular information attributes. For example, hues may be applied to emphasize the hierarchy structure. InfoShape needs more interactive functionalities. The current InfoShape provides users an overview of multi-dimensional information, so that users can quickly understand the similarities and the difference between different multi-dimensional

information sets. However, it lacks some necessary interactions, such as functions that help the user with drilling down from high level to detail level. One major limitation needing to be improved for SciTrend is that users cannot see the trend of a computer science area over time and its connections to other research areas at the same time.

APPENDIX A

JOURNALS USED IN SCITREND PROJECT

COMPUTER SCIENCE RESEARCH AREA	JOURNAL NAME
COMPUTER SCIENCE, THEORY & METHODS	ACM COMPUT SURV
COMPUTER SCIENCE, THEORY & METHODS	ACM T COMPUT LOG
COMPUTER SCIENCE, THEORY & METHODS	ACM T COMPUT SYST
COMPUTER SCIENCE, THEORY & METHODS	ADV MATH COMMUN
COMPUTER SCIENCE, THEORY & METHODS	COMB PROBAB COMPUT
COMPUTER SCIENCE, THEORY & METHODS	COMPUT COMPLEX
COMPUTER SCIENCE, THEORY & METHODS	COMPUTING
COMPUTER SCIENCE, THEORY & METHODS	CRYPTOLOGIA
COMPUTER SCIENCE, THEORY & METHODS	DESIGN CODE CRYPTOGR
COMPUTER SCIENCE, THEORY & METHODS	DISCRETE COMPUT GEOM
COMPUTER SCIENCE, THEORY & METHODS	DISTRIB COMPUT
COMPUTER SCIENCE, THEORY & METHODS	FORM METHOD SYST DES
COMPUTER SCIENCE, THEORY & METHODS	FOUND COMPUT MATH
COMPUTER SCIENCE, THEORY & METHODS	FUTURE GENER COMP SY
COMPUTER SCIENCE, THEORY & METHODS	FUZZY SET SYST
COMPUTER SCIENCE, THEORY & METHODS	IEEE ANN HIST COMPUT
COMPUTER SCIENCE, THEORY & METHODS	IEEE T INF FOREN SEC
COMPUTER SCIENCE, THEORY & METHODS	IEEE T PARALL DISTR
COMPUTER SCIENCE, THEORY & METHODS	INFORM COMPUT
COMPUTER SCIENCE, THEORY & METHODS	INT J COMPUT GEOM AP
COMPUTER SCIENCE, THEORY & METHODS	INT J FOUND COMPUT S
COMPUTER SCIENCE, THEORY & METHODS	INT J GEN SYST
COMPUTER SCIENCE, THEORY & METHODS	INT J PARALLEL PROG
COMPUTER SCIENCE, THEORY & METHODS	INT J QUANTUM INF
COMPUTER SCIENCE, THEORY & METHODS	INT J SYST SCI
COMPUTER SCIENCE, THEORY & METHODS	J ALGORITHM
COMPUTER SCIENCE, THEORY & METHODS	J CELL AUTOM
COMPUTER SCIENCE, THEORY & METHODS	J COMPLEXITY
COMPUTER SCIENCE, THEORY & METHODS	J COMPUT ANAL APPL
COMPUTER SCIENCE, THEORY & METHODS	J CRYPTOL
COMPUTER SCIENCE, THEORY & METHODS	J LOGIC ALGEBR PROGR
COMPUTER SCIENCE, THEORY & METHODS	J PARALLEL DISTR COM

COMPUTER SCIENCE, THEORY & METHODS	J LOGIC COMPUT
COMPUTER SCIENCE, THEORY & METHODS	J SYMB COMPUT
COMPUTER SCIENCE, THEORY & METHODS	J ZHEJIANG UNIV-SC A
COMPUTER SCIENCE, THEORY & METHODS	MATH STRUCT COMP SCI
COMPUTER SCIENCE, THEORY & METHODS	MULTIDIM SYST SIGN P
COMPUTER SCIENCE, THEORY & METHODS	PARALLEL COMPUT
COMPUTER SCIENCE, THEORY & METHODS	QUANTUM INF COMPUT
COMPUTER SCIENCE, THEORY & METHODS	REAL-TIME SYST
COMPUTER SCIENCE, THEORY & METHODS	SIAM J COMPUT
COMPUTER SCIENCE, THEORY & METHODS	STAT COMPUT
COMPUTER SCIENCE, THEORY & METHODS	THEOR COMPUT SCI
COMPUTER SCIENCE, THEORY & METHODS	THEOR COMPUT SYST

COMPUTER SCIENCE, SOFTWARE ENGINEERING	ACM SIGPLAN NOTICES
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ACM T GRAPHIC
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ACM T MATH SOFTWARE
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ACM T PROGR LANG SYS
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ACM T SOFTW ENG METH
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ALGORITHMICA
COMPUTER SCIENCE, SOFTWARE ENGINEERING	BIT
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT AIDED DESIGN
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT AIDED GEOM D
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT ANIMAT VIRT W
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT GRAPH FORUM
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT GRAPH WORLD
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT GRAPH-UK
COMPUTER SCIENCE, SOFTWARE ENGINEERING	COMPUT LANG SYST STR
COMPUTER SCIENCE, SOFTWARE ENGINEERING	CONCURR COMP-PRACT E
COMPUTER SCIENCE, SOFTWARE ENGINEERING	DISCRETE MATH THEOR
COMPUTER SCIENCE, SOFTWARE ENGINEERING	DR DOBBS J
COMPUTER SCIENCE, SOFTWARE ENGINEERING	EMPIR SOFTW ENG
COMPUTER SCIENCE, SOFTWARE ENGINEERING	FORM ASP COMPUT
COMPUTER SCIENCE, SOFTWARE ENGINEERING	FUND INFORM
COMPUTER SCIENCE, SOFTWARE ENGINEERING	GRAPH MODELS
COMPUTER SCIENCE, SOFTWARE ENGINEERING	ICGA J
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEE P-SOFTW
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEEE COMPUT GRAPH
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEEE INTERNET COMPUT
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEEE SOFTWARE
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEEE T SOFTWARE ENG
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IEEE T VIS COMPUT GR
COMPUTER SCIENCE, SOFTWARE ENGINEERING	IET SOFTW
COMPUTER SCIENCE, SOFTWARE ENGINEERING	INT J ELECTRON COMM

COMPUTER SCIENCE, SOFTWARE ENGINEERING	INT J WAVELETS MULTI
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J FUNCT PROGRAM
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J SOFTW MAINT EVOL-R
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J SYST SOFTWARE
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J UNIVERS COMPUT SCI
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J VISUAL LANG COMPUT
COMPUTER SCIENCE, SOFTWARE ENGINEERING	J WEB ENG
COMPUTER SCIENCE, SOFTWARE ENGINEERING	MATH PROGRAM
COMPUTER SCIENCE, SOFTWARE ENGINEERING	OPTIM METHOD SOFTW
COMPUTER SCIENCE, SOFTWARE ENGINEERING	PROGRAM COMPUT SOFT+
COMPUTER SCIENCE, SOFTWARE ENGINEERING	RANDOM STRUCT ALGOR
COMPUTER SCIENCE, SOFTWARE ENGINEERING	SCI COMPUT PROGRAM
COMPUTER SCIENCE, SOFTWARE ENGINEERING	SOFTW TEST VERIF REL
COMPUTER SCIENCE, SOFTWARE ENGINEERING	SOFTWARE PRACT EXPER
COMPUTER SCIENCE, SOFTWARE ENGINEERING	SOFTWARE QUAL J
COMPUTER SCIENCE, SOFTWARE ENGINEERING	THEOR PRACT LOG PROG
COMPUTER SCIENCE, SOFTWARE ENGINEERING	VISUAL COMPUT

COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ACM T MODEL COMPUT S
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ADV ENG SOFTW
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	APPL ALGEBR ENG COMM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ARCH COMPUT METHOD E
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	CIN-COMPUT INFORM NU
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPEL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT APPL ENG EDUC
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT BIOL CHEM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT BIOL MED
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT CHEM ENG
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT CONCRETE
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT EDUC
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT ELECTRON AGR
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT FLUIDS
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT GEOSCI-UK
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT GEOTECH

COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT IND
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT IND ENG
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT MATH APPL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT METH PROG BIO
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT METHOD BIOMEC
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT MUSIC J
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT OPER RES
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT PHYS COMMUN
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT SCI ENG
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT STAT DATA AN
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT STRUCT
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUT-AIDED CIV INF
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	COMPUTAT GEOSCI
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	CONCURRENT ENG-RES A
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ENG COMPUT-GERMANY
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ENG COMPUTATION
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ENVIRON MODELL SOFTW
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	IEEE ACM T COMPUT BI
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	IEEE T IND INFORM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	IEEE T MED IMAGING
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	IND MANAGE DATA SYST
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	INFORMS J COMPUT
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	INT J COMPUT INTEG M
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	INT J MOD PHYS C
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	INT J NUMER METH FL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	INT J RF MICROW C E
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J BIOMED INFORM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMB OPTIM

COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT AID MOL DES
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT BIOL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT CIVIL ENG
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT INF SCI ENG
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT PHYS
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J COMPUT-AIDED MATER
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J HYDROINFORM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J MOL GRAPH MODEL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J MOL MODEL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J NEW MUSIC RES
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J STAT COMPUT SIM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J STAT SOFTW
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	J VISUAL-JAPAN
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	LANG RESOUR EVAL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	MATCH-COMMUN MATH CO
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	MATH COMP MODEL DYN
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	MATH COMPUT MODEL
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	MATH COMPUT SIMULAT
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	MED BIOL ENG COMPUT
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	NEUROINFORMATICS
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	QSAR COMB SCI
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	QUEUEING SYST
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	ROBOT CIM-INT MANUF
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SAR QSAR ENVIRON RES
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SCIENTOMETRICS
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SIMUL MODEL PRACT TH
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SIMUL-T SOC MOD SIM
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SOC SCI COMPUT REV

COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	SPEECH COMMUN
COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS	STRUCT MULTIDISCIP O
COMPUTER SCIENCE, INFORMATION SYSTEMS	ACM T DATABASE SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	ACM T INFORM SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	ACM T MULTIM COMPUT
COMPUTER SCIENCE, INFORMATION SYSTEMS	ACTA INFORM
COMPUTER SCIENCE, INFORMATION SYSTEMS	ANNU REV INFORM SCI
COMPUTER SCIENCE, INFORMATION SYSTEMS	ASLIB PROC
COMPUTER SCIENCE, INFORMATION SYSTEMS	BELL LABS TECH J
COMPUTER SCIENCE, INFORMATION SYSTEMS	CLUSTER COMPUT
COMPUTER SCIENCE, INFORMATION SYSTEMS	COMPUT COMMUN
COMPUTER SCIENCE, INFORMATION SYSTEMS	COMPUT COMMUN REV
COMPUTER SCIENCE, INFORMATION SYSTEMS	COMPUT SECUR
COMPUTER SCIENCE, INFORMATION SYSTEMS	DIGIT INVEST
COMPUTER SCIENCE, INFORMATION SYSTEMS	DISTRIB PARALLEL DAT
COMPUTER SCIENCE, INFORMATION SYSTEMS	ELECTRON COMMER R A
COMPUTER SCIENCE, INFORMATION SYSTEMS	EUR J INFORM SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	GEOINFORMATICA
COMPUTER SCIENCE, INFORMATION SYSTEMS	IBM SYST J
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE PERVAS COMPUT
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE SECUR PRIV
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE T INF TECHNOL B
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE T INFORM THEORY
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE T MOBILE COMPUT
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEEE T MULTIMEDIA
COMPUTER SCIENCE, INFORMATION SYSTEMS	IEICE T INF SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	IET INFORM SECUR
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFOR
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM MANAGE-AMSTER
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM PROCESS LETT
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM PROCESS MANAG
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM RETRIEVAL
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM SCIENCES
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM SOFTWARE TECH
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM SYST FRONT
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM SYST MANAGE
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORM TECHNOL LIBR
COMPUTER SCIENCE, INFORMATION SYSTEMS	INFORMATICA-LITHUAN
COMPUTER SCIENCE, INFORMATION SYSTEMS	INT J COOP INF SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	INT J DISTRIB SENS N

COMPUTER SCIENCE, INFORMATION SYSTEMS	INT J GEOGR INF SCI
COMPUTER SCIENCE, INFORMATION SYSTEMS	INT J MED INFORM
COMPUTER SCIENCE, INFORMATION SYSTEMS	INT J WEB SERV RES
COMPUTER SCIENCE, INFORMATION SYSTEMS	INTERNET RES
COMPUTER SCIENCE, INFORMATION SYSTEMS	J AM MED INFORM ASSN
COMPUTER SCIENCE, INFORMATION SYSTEMS	J AM SOC INF SCI TEC
COMPUTER SCIENCE, INFORMATION SYSTEMS	J ASSOC INF SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	J CHEM INF MODEL
COMPUTER SCIENCE, INFORMATION SYSTEMS	J COMMUN NETW-S KOR
COMPUTER SCIENCE, INFORMATION SYSTEMS	J DATABASE MANAGE
COMPUTER SCIENCE, INFORMATION SYSTEMS	J INF SCI
COMPUTER SCIENCE, INFORMATION SYSTEMS	J INF SCI ENG
COMPUTER SCIENCE, INFORMATION SYSTEMS	J INF TECHNOL
COMPUTER SCIENCE, INFORMATION SYSTEMS	J MANAGE INFORM SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	J NETW SYST MANAG
COMPUTER SCIENCE, INFORMATION SYSTEMS	J ORG COMP ELECT COM
COMPUTER SCIENCE, INFORMATION SYSTEMS	J RES PRACT INF TECH
COMPUTER SCIENCE, INFORMATION SYSTEMS	J SIGNAL PROCESS SYS
COMPUTER SCIENCE, INFORMATION SYSTEMS	J STRATEGIC INF SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	J VIS COMMUN IMAGE R
COMPUTER SCIENCE, INFORMATION SYSTEMS	J VLSI SIG PROC SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	MED INFORM INTERNET
COMPUTER SCIENCE, INFORMATION SYSTEMS	METHOD INFORM MED
COMPUTER SCIENCE, INFORMATION SYSTEMS	MIS QUART
COMPUTER SCIENCE, INFORMATION SYSTEMS	MULTIMED TOOLS APPL
COMPUTER SCIENCE, INFORMATION SYSTEMS	MULTIMEDIA SYST
COMPUTER SCIENCE, INFORMATION SYSTEMS	ONLINE INFORM REV
COMPUTER SCIENCE, INFORMATION SYSTEMS	OPEN SYST INF DYN
COMPUTER SCIENCE, INFORMATION SYSTEMS	PERS UBIQUIT COMPUT
COMPUTER SCIENCE, INFORMATION SYSTEMS	PHOTONIC NETW COMMUN
COMPUTER SCIENCE, INFORMATION SYSTEMS	PROGRAM-ELECTRON LIB
COMPUTER SCIENCE, INFORMATION SYSTEMS	RAIRO-THEOR INF APPL
COMPUTER SCIENCE, INFORMATION SYSTEMS	REQUIR ENG
COMPUTER SCIENCE, INFORMATION SYSTEMS	SCI CHINA SER F
COMPUTER SCIENCE, INFORMATION SYSTEMS	SIGMOD REC
COMPUTER SCIENCE, INFORMATION SYSTEMS	WIREL COMMUN MOB COM
COMPUTER SCIENCE, INFORMATION SYSTEMS	WIREL NETW
COMPUTER SCIENCE, INFORMATION SYSTEMS	WIRTSCHAFTSINF
COMPUTER SCIENCE, INFORMATION SYSTEMS	WORLD WIDE WEB
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	ACM T DES AUTOMAT EL
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	ADV COMPUT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	ANALOG INTEGR CIRC S

COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	CAN J ELECT COMPUT E
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMMUN ACM
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUT ELECTR ENG
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUT J
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUT NETW
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUT STAND INTER
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUT SYST SCI ENG
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	COMPUTER
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	DES AUTOM EMBED SYST
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	DISPLAYS
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IBM J RES DEV
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE ACM T NETWORK
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE DES TEST COMPUT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE MICRO
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE MULTIMEDIA
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE NETWORK
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE T COMPUT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE T COMPUT AID D
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE T DEPEND SECURE
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE T RELIAB
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE T VLSI SYST
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEEE WIREL COMMUN
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IEICE T FUND ELECTR
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	IET COMPUT DIGIT TEC
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	INT J HIGH PERFORM C
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	INTEGRATION
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J ACM
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J CIRCUIT SYST COMP
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J COMPUT SCI TECHNOL
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J COMPUT SYST SCI
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J HIGH SPEED NETW
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J NETW COMPUT APPL
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J OPT NETW
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J SUPERCOMPUT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	J SYST ARCHITECT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	MICROPROCESS MICROSY
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	MOBILE NETW APPL
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	NETWORKS
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	NEW GENERAT COMPUT
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	PERFORM EVALUATION
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE	VLDB J
COMPUTER SCIENCE, CYBERNETICS	BEHAV INFORM TECHNOL

COMPUTER SCIENCE, CYBERNETICS	BIOL CYBERN
COMPUTER SCIENCE, CYBERNETICS	CONTROL CYBERN
COMPUTER SCIENCE, CYBERNETICS	CYBERNET SYST
COMPUTER SCIENCE, CYBERNETICS	HUM-COMPUT INTERACT
COMPUTER SCIENCE, CYBERNETICS	IEEE T SYST MAN CY A
COMPUTER SCIENCE, CYBERNETICS	INT J HUM-COMPUT INT
COMPUTER SCIENCE, CYBERNETICS	INT J HUM-COMPUT ST
COMPUTER SCIENCE, CYBERNETICS	INTERACT COMPUT
COMPUTER SCIENCE, CYBERNETICS	KYBERNETES
COMPUTER SCIENCE, CYBERNETICS	KYBERNETIKA
COMPUTER SCIENCE, CYBERNETICS	PRESENCE-TELEOP VIRT
COMPUTER SCIENCE, CYBERNETICS	USER MODEL USER-ADAP

COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ADAPT BEHAV
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ADV ENG INFORM
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	AI COMMUN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	AI EDAM
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	AI MAG
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ANN MATH ARTIF INTEL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	APPL ARTIF INTELL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	APPL INTELL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	APPL SOFT COMPUT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ARTIF INTELL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ARTIF INTELL MED
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ARTIF INTELL REV
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ARTIF LIFE
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	AUTON AGENT MULTI-AG
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	AUTON ROBOT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	CHEMOMETR INTELL LAB
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COGN SYST RES
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COMPUT INFORM
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COMPUT INTELL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COMPUT LINGUIST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COMPUT SPEECH LANG
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	COMPUT VIS IMAGE UND
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	CONNECT SCI
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	CONSTRAINTS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	DATA KNOWL ENG
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	DATA MIN KNOWL DISC
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	DECIS SUPPORT SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ENG APPL ARTIF INTEL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ENG INTELL SYST ELEC
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	EVOL COMPUT

COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	EXPERT SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	EXPERT SYST APPL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE COMPUT INTELL M
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE INTELL SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T EVOLUT COMPUT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T FUZZY SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T IMAGE PROCESS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T KNOWL DATA EN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T NEURAL NETWOR
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T PATTERN ANAL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T SYST MAN CY B
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IEEE T SYST MAN CY C
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IET COMPUT VIS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	IMAGE VISION COMPUT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INFORM FUSION
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J APPROX REASON
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J ARTIF INTELL T
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J COMPUT VISION
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J DOC ANAL RECOG
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J INF TECH DECIS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J INNOV COMPUT I
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J INTELL SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J NEURAL SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J PATTERN RECOGN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J SEMANT WEB INF
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J SOFTW ENG KNOW
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INT J UNCERTAIN FUZZ
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INTEGR COMPUT-AID E
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INTELL AUTOM SOFT CO
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	INTELL DATA ANAL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J ARTIF INTELL RES
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J AUTOM REASONING
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J CHEMOMETR
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J COMPUT SYS SC INT+
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J EXP THEOR ARTIF IN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J HEURISTICS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J INTELL FUZZY SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J INTELL INF SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J INTELL MANUF
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J INTELL ROBOT SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J MACH LEARN RES
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J MATH IMAGING VIS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J MULT-VALUED LOG S

COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	J WEB SEMANT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	KNOWL ENG REV
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	KNOWL INF SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	KNOWL-BASED SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	MACH LEARN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	MACH VISION APPL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	MECHATRONICS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	MED IMAGE ANAL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	MIND MACH
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NETWORK-COMP NEURAL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEURAL COMPUT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEURAL COMPUT APPL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEURAL NETW WORLD
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEURAL NETWORKS
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEURAL PROCESS LETT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	NEUROCOMPUTING
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	PATTERN ANAL APPL
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	PATTERN RECOGN
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	PATTERN RECOGN LETT
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	ROBOT AUTON SYST
COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE	SOFT COMPUT

APPENDIX B

EXAMPLE LAYOUTS OF IMPROVED JCR CITATION PATTERNS (1999-2007)

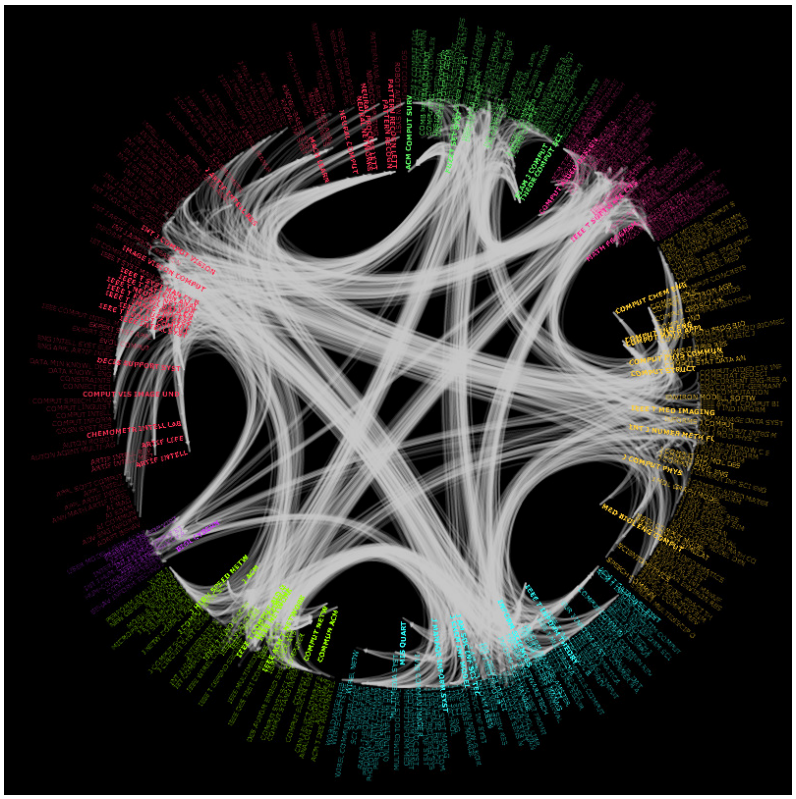


Figure B.1. Improved JCR Citation Pattern Layout on 1999 Citation Data

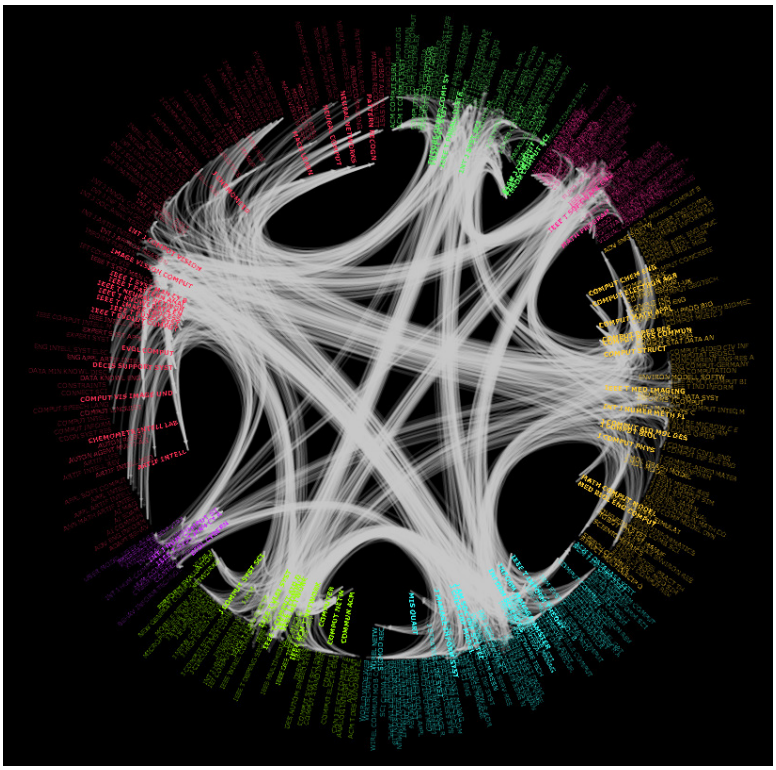


Figure B.2. Improved JCR Citation Pattern Layout on 2000 Citation Data

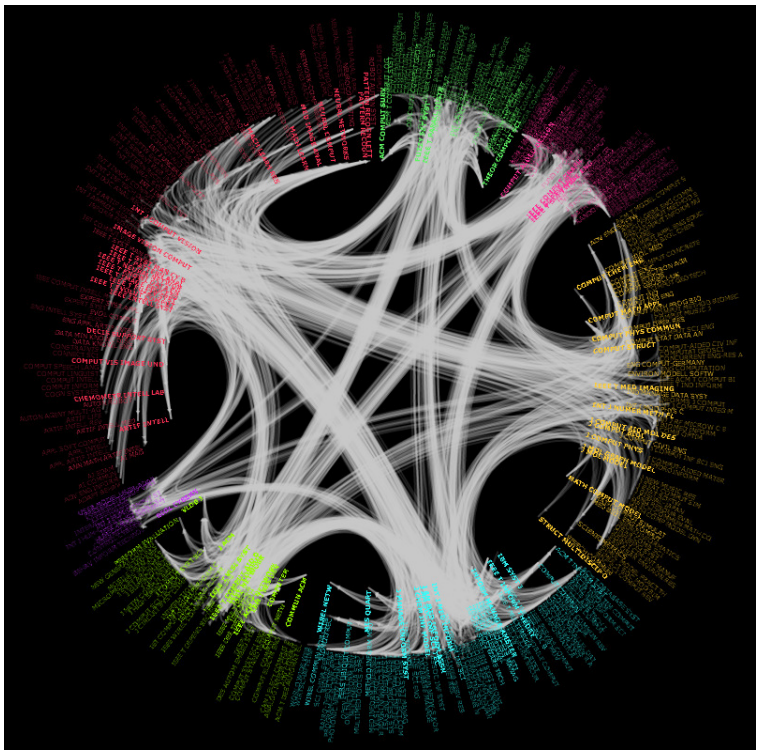


Figure B.3. Improved JCR Citation Pattern Layout on 2001 Citation Data

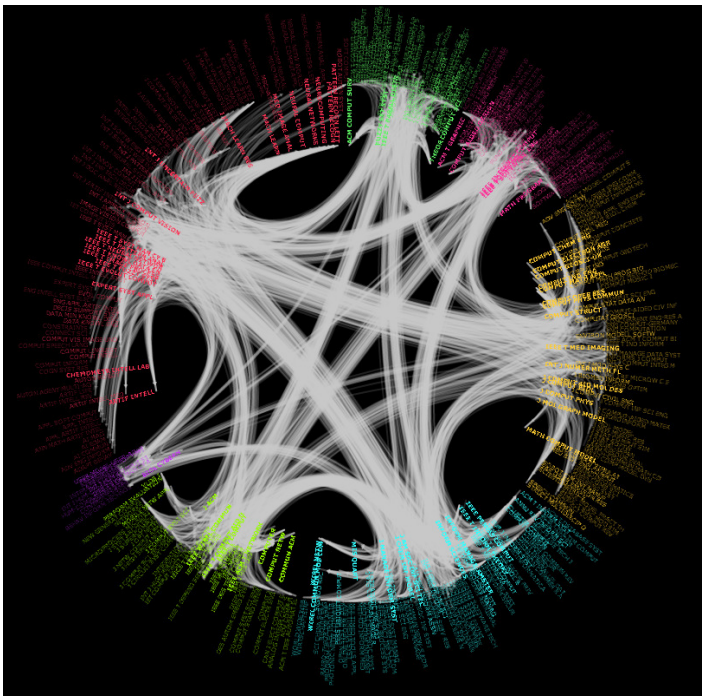


Figure B.4. Improved JCR Citation Pattern Layout on 2002 Citation Data

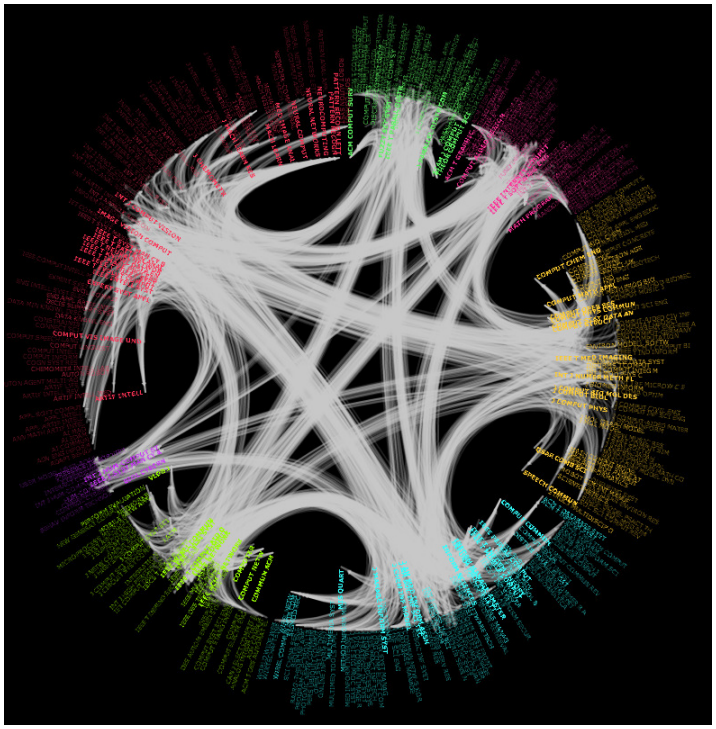


Figure B.5. Improved JCR Citation Pattern Layout on 2003 Citation Data

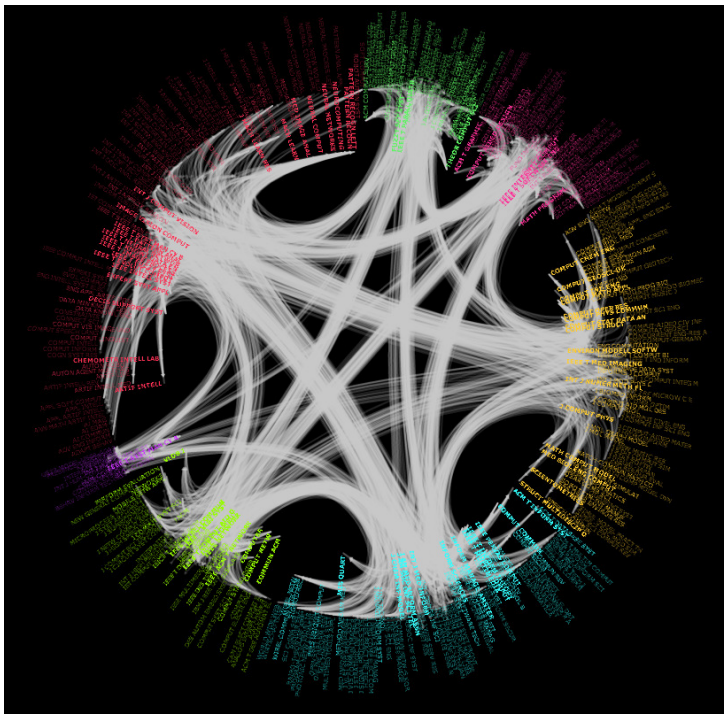


Figure B.6. Improved JCR Citation Pattern Layout on 2004 Citation Data

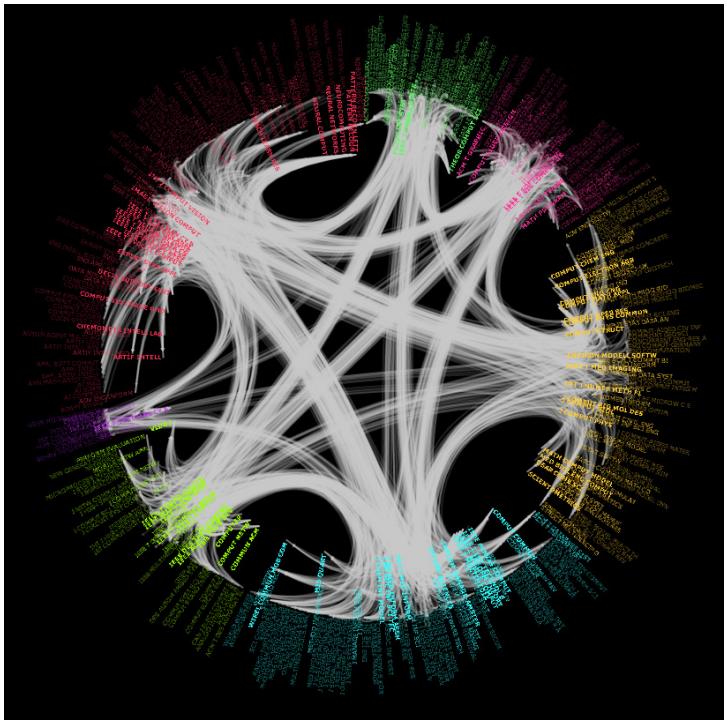


Figure B.7. Improved JCR Citation Pattern Layout on 2005 Citation Data

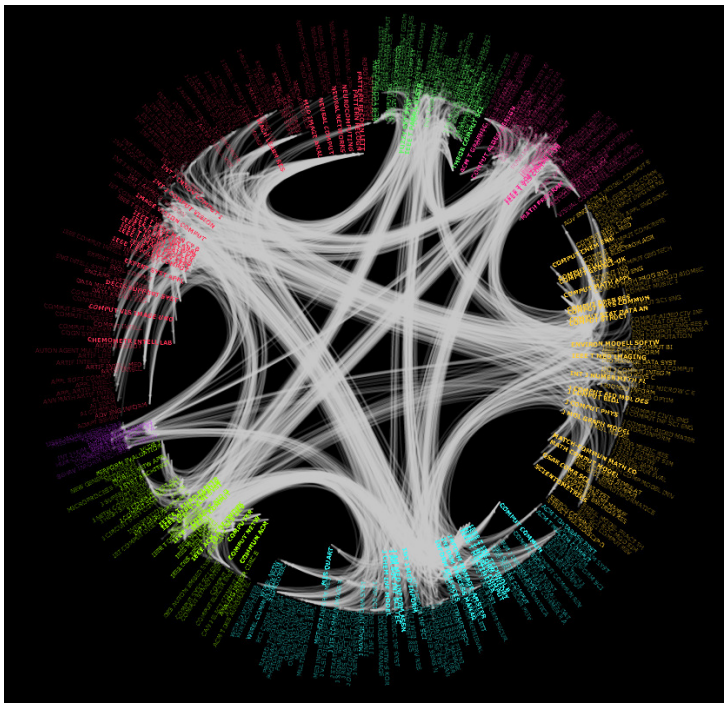


Figure B.8. Improved JCR Citation Pattern Layout on 2006 Citation Data

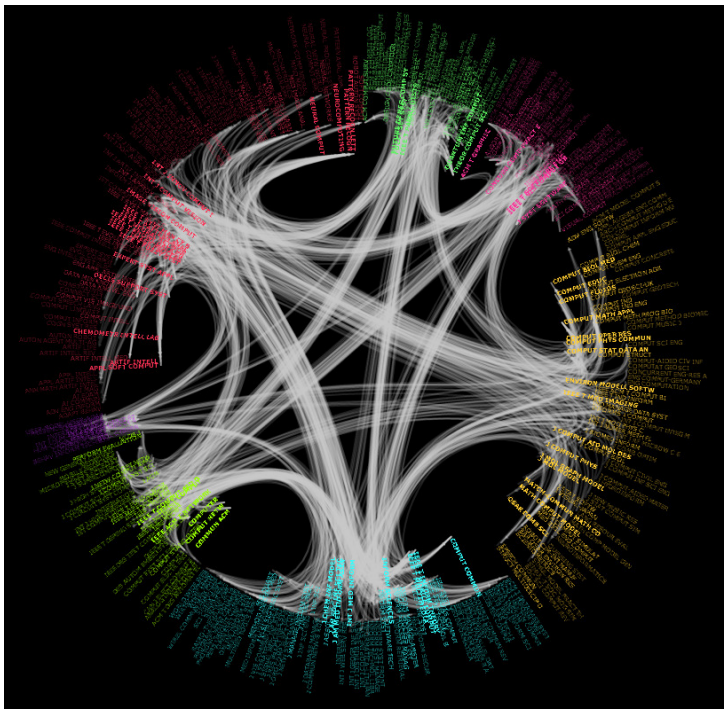


Figure B.9. Improved JCR Citation Pattern Layout on 2007 Citation Data

BIBLIOGRAPHY

[1] J. Abello and J. Korn, MGV: A System for Visualizing Massive Multi-digraphs, *IEEE Transcations. on Visualization and Computer Graphics*, Volume: 8, Number: 1, Page(s): 21-38, 2002.

[2] D.F. Andrews, Plots of High-Dimensional Data, *Biometrics*, Volume: 28, Page(s): 125-136, 1972.

[3] K. Andrew, Visualising cyberspace: Information visualisation in the Harmony internet browser, *The proceedings of IEEE Information Visualization '95*, IEEE Computer Press, Los Alamitos, CA, Page(s): 97-104, 1995

[4] K. Andrews. *Human-Computer Interaction*, Lecture Notes (2006 version), Web site: http://courses.iicm.edu/hci/hci.pdf.

[5] K. Andrews, Evaluating information visualisations, *The Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, Page(s): 1-5, May 23-23, 2006, Venice, Italy.

[6] K. Andrews and H. Heidegger, Information Slices: Visualising and Exploring Large Hierarchy Using Cascading, Semi-circular Discs, *The Procceedings of IEEE Symposium on Information Visualization*, Page(s): 9-12, 1998.

[7] K. Andrews, W. Putz, and A. Nussbaumer, The Hierarchy Visualization System, *Information Visualization 2007 (IV 2007)*, Page(s): 257-262, 2007.

[8] M. Ankerst, D.A. Keim, and H.P. Kriegel, Circle Segments: A Technique for Visually Exploring Large Multi-dimensional Data Sets, *The Proceedings of the 7th conference on Visualization 1996*, Hot Topic Session, San Francisco, CA, 1996.

[9] Y. Arase, T. Hara, T. Uemukai, and S. Nishio, OPA Browser: A Web Browser for Cellular Phone Users, *The Proceedings of the 20th annual ACM symposium on User Interface Software and Technology*, Newport, USA, Page(s): 71-80, 2007.

Y. Arase, T. Hara, T. Uemukai, and S. Nishio, OPA Browser: A Web Browser for Cellular Phone Users, *The Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, Newport, USA, Page(s): 71-80, 2007.

[10] E. Arias, L.R. Curtin, R. Wei and R.N. Anderson, United States Decennial Life Tables for 1999–2001, United States Life Tables, *National Vital Statistics Reports*, Volume: 57 Unmber: 1, Hyattsville, MD: National Center for Health Statistics. 2008.

[11] M. Balzer and O. Deussen, Voronoi Treemaps, *The Procceddings. of the 2005 IEEE Symposium on Information Visualization*, Page(s): 7, 2005.

[12] G.D. Battista, P. Eades, R. Tamassia, and I. Tollis: Annotated Bibliography on Graph Drawing Algorithms, Computational Geometry, *Theory and Applications*, Volume: 4, Page(s): 235-282, 1994.

[13] P. Baudisch, X. Xie, C. Wang, and W-Y. Ma, Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content, *The Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, Santa Fe, USA, Page(s): 91-94, 2004.

[14] J. Beddow, Shape Coding of Multi-dimensional Data on a Mircocomputer Display, Proceedings *of the 1st conference on Visualization 1990*, San Francisco, CA, Page(s). 238-246, 1990.

[15] P. Bergstrom, E.J. Whitehead Jr., CircleView: Scalable Visualization and Navigation of Citation Networks, *The Proceedings, of the 2006 Symposium on Interactive Visual Information Collections and Activity (IVICA)*, College Station, 2006.

[16] R. Boardman, Bubble Trees: The Visualization of Hierarchical Information Structures, *Extended Abstracts on Human Factors in Computing Systems (CHI 2000)*, ACM Press, The Hague, The Netherlands, Page(s). 315–316, 2000.

[17] C.M. Brown, *Human-Computer Interface Design Guidelines*, Ablex, NJ, 1988. ISBN 0893913324 (com, uk).

[18] C. Buchheim, M. Junger, and S. Leipert, Improving Walker's Algorithm to Run in Linear Time, *Revised Papers from the 10th International Symposium. On Graph Drawing (GD 2002)*. Springer, Irvine, California, USA, Page(s): 344-353, 2002.

[19] T. Bürger, Magic Eye View: Eine neue Fokus + Kontext Technik zur Darstellung von Graphen. *Master's Thesis*, Institute for Computer Graphics, University of Rostock, Germany (1999).

[20] M.D. Byrne, B.E. John, and E. Joyce, *A Day in the Life of Ten WWW Users*, 2000, http://citeseer.ist.psu.edu/400156.html. Unpublished paper.

[21] M.D. Byrne, B.E. John, N.S.Wehrle and D.C. Crow, The Tangled Web We Wove: A Taskonomy of WWW Use, *The Proccedings of. Conference on Human Factors in Computing Systems (CHI 1999),* Page(s): 544–551, ACM, Pittsburgh, Pennsylvania, USA, May 1999.

[22] S.K. Card, A. Newell and T.P. Moran, *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, NJ, USA, ISBN 0898598591 (com, uk), (Cited on page 96.), 1983.

[23] S.K. Card, J.D. Mackinlay, and B. Shneiderman, *Reading in Information Visualization – Using Vision to Think*, The Morgan Kaufmann series in interactive technologies, Mogan Kaufmann, 1999.

[24] I. Ceaparu, J. Lazar, K. Bessiere, J. Robinson, and B. Shneiderman, Determining Causes and Severity of End-User Frustration, *Technical Report CS-TR-4371,* University of Maryland, Computer Science Department, May 2003.

[25] C. Chen, *Information Visualization and Virtual Environments*, Springer-Verlag, London, 1999.

[26] H. Chernoff, The Use of Faces to Represent Points in k-Dimensional Space Graphically, *Journal Amer. Statistical Association*, Volume: 68, Page(s): 361-368, 1971.

[27] E.H. Chi, A Taxonomy of Visualization Techniques Using the Data State Reference Model, *The Procceedings of Symposium Information Visualization (InfoVis)*, Page(s): 69-75, 2000.

[28] E.H. Chi, and J.T. Riedl, An Operator Interaction Framework for Visualization Systems, *The Procceedings of Symposium in Information Visualization (InfoVis)*, Page(s): 63-70, 1998.

[29] K.W. Church, and J.I. Helfman, Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code, *The Proceedings of the 24th Symposium on the Interface, Computing Science and Statistics*, Volume: 24, Page(s): 58-67, March, 1992.

[30] W.S. Cleveland, *Visualizing Data*, AT&T Bell Laboratories, Murray Hill, NJ, Hobart Press, Summit NJ, 1993.

[31] P.J. Denning, The ACM Digital Library Goes Live, *Communication of the ACM*, Volume: 40, Issue: 7, Page(s): 28-29, 1997

[32] G. DiBattista, P. Eades, R. Tamassia, and I.G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.

[33] M. Dontcheva, S.M. Drucker, G. Wade, D. Salesin, and M.F. Cohen, Summarizing Personal Web Browsing Sessions, *The Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, Montreux, Switzerland, Page(s). 115-124, 2006.

[34] N. Elmqvist and P. Tsigas, CiteWiz: a Tool for the Visualization of Scientific Citation Networks, *Information Visualization*, Volume: 6, Issue: 3, Page(s): 215-232, 2007.

[35] N. Elmqvist and P. Tsigas, CiteWiz: a Tool for the Visualization of Scientific Citation Networks, Information Visualization, *Technical Report*, Department of Computer Science, Chalmers University of Technology and Goteborg University, Sweden 2004.

[36] N. Elmqvist and P. Tsigas, Causality Visualization using Animated Growing Polygons, *The Proceedings of the IEEE Symposium on Information Visualization*, Page(s): 189–196, 2003.

[37] Q. Du, V. Faber, and M. Gunzburger, Centroidal Voronoi Tessellations: Applications and Algorithms, *SIAM Review*, Volume: 41, Issue: 4, Page(s): 637– 676, 1999.

[38] P. Eades, *Drawing Free Trees*, Bulletin of the Institute for Combinatorics and its Applications, Volume: 5, Number: 2, Page(s):10–36, 1992.

[39] S. Feiner and C. Beshers, World within World: Metaphors for Exploring n-dimensional Virtual Worlds, *The Proceedings of the 3th Annual ACM Symposium on User Interface Software and Technology*, Page(s) 76-83, 1990.

[40] A.P. Field and G.J. Hole, *How to Design and Report Experiments*. Sage Publications (CA), 2002.

[41] G.W. Furnas and B.B. Bederso, Space-Scale Diagrams: Understanding Multiscale Interfaces, *The Proc. Conference on Human Factors in Computing Systems (CHI 1995)*, Page(s): 234-241, 1995.

[42] E. Garfield, *Citation Indexing - Its Theory and Application in Science*, Technology and Humanities Philadelphia ISI Press, 1983.

[43] E. Garfield, Citation Analysis as a Tool in Journal Evaluation, *Science*, Page(s): 471-479, 1972

[44] N. Gershon, S.G. Eick, and S. Card, Information Visualization, *ACM Interactions*, Volume: 5, Issue: 2, Page(s): 9–15.

[45] C.L. Giles, K.D. Bollacker, S. Lawrence, CiteSeer: An automatic Citation Indexing System, *The Procceedings of the Third ACM Confference on Digital Libraries*, Page(s): 89-98, Pittsurgh, Pennsylvania, USA, 1998

[46] J. Halverston, *The First Pictures, Perceptual Foundations of Paleolithic Art*, Perception 21, Page(s): 389-404, 1992.

[47] J. Hao, K. Zhang, and T.C. Hsieh, A Mobile Interface for Hierarchical Information Visualization and Navigation, *Proc. 11th Annual IEEE Symposium on Consumer Electronics (ISCE'07)*, Dallas, USA, 20-23 June 2007.

[48] J. Hao, K. Zhang, and M.L. Huang, RELT: Visualizing Trees on Mobile Devices, *In Lecture Notes in Computer Science (LNCS), VISUAL2007*, vol. 4781; pages 344-357, Springer-Verlag, 2007.

[49] J. Hao, K. Zhang, and Q.M. Zhu, Managing Hierarchical Information on Small Screens, *International Conferences on Asia-Pacific Web Conference and Web-Age Information Management (APWeb/WAIM)*, Suzhou, China, 1-4 Apr 2009.

[50] J. Hao, C.A. Gabrysch, C.Y. Zhao, Q.M. Zhu, K. Zhang, Visualizing and Navigating Hierarchical Information on Mobile User Interfaces, *International Journal of Advanced Intelligence, AIA,* 2010 (Accepted).

[51] J. Helfman, Dotplot Patterns: a Literal Look at Pattern Languages, *Theory and Practice of Object Systems*, Volume: 2, Number: 1, Page(s): 31-41, 1996.

[52] R. J. Hendley, N.S. Drew, and A.S. Wood, Narcissus: Visualizing information, *Proceedings of IEEE Information Visualization '95*, IEEE Computer Press, Los Alamitos, CA, Page(s): 90-96, 1995.

[53] I. Herman, G. Melançon, and M.S. Marshall, Graph Visualization in Information Visualization: a Survey, *The IEEE Transcations on Visualization and Computer Graphics*, Page(s). 24-44, 2000.

[54] I. Herman, G. Melancon, M.M. de Ruiter, and M. Delest, Latour: A Tree Visualisation System, *Symposium on Graph Drawing (GD'99)*, Springer-Verlag, Stirin Castle, Czech Republic, Page(s): 392-399, 1999.

[55] E. Hetzler and A. Turner. Analysis Experiences Using Information Visualization. *IEEE Computer Graphics and Applications*, Volume: 24, Issue: 5, Page(s): 22–26, IEEE Computer Society Press, Los Alamitos, CA, USA September/October 2004.

[56] D. Holten, Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchy Data, *The IEEE Transctions on Visualization and Computer Graphics*, Volume: 12, Issue: 5, Page(s): 741-748, 2006

[57] D.M. Hilbert and D.F. Redmiles. Extracting Usability Information from User Interface Events, *ACM Computing Surveys (CSUR)*, Volume: 32, Issue: 4, Page(s): 384–421, December 2000.

[58] P.E. Hoffman, *Table Visualizations: A Formal Model and Its Applications*, Doctoral Dissertation, Computer Science Dept., Univ. of Massachusetts at Lowell, 1999.

[59] A. Inselberg: The Plane with Parallel Coordinates, Special Issue on Computational Geometry, *The Visual Computer*, Volume: 1, Page(s). 69-97, 1985.

[60] A. Inselberg, B. Dimsdale: Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry, *The Proccedings of 1st Conference of Visualization*, San Francisco, CA, 1990, Page(s): 361-370, 1990.

[61] T.J. Jankun-Kelly and K.L. Ma, MoireGraphs: Radial Focus plus Context Visualization and Interaction for Graphs with Visual Nodes, *The IEEE Symposium. on Information Visualization 2003*, IEEE Computer Society, Seattle, Washington, USA, Page(s): 59-66, 2003.

[62] H.J. Jeffrey, Chaos Game Representation of Gene Structure, *Nucleic Acids Research*, Volume: 18, Number: 8, 2163-2170, 1990.

[63] C. S Jeong and A. Pang, Reconfigurable Disc Trees for Visualizing Large Hierarchical Information Space, *The Procceddings in. IEEE Symposium on Information Visualization*, Page(s): 19-25, 1998.

[64] B.E. John and H.Packer, Learning and Using the Cognitive Walkthrough Method: A Case Study Approach, *The Proccedings of Conference on Human Factors in Computing Systems (CHI 1995)*, Page(s): 429-436, Denver, Colorado, USA, May 1995.

[65] M.S. John, M.S. Cowen, H.S. Smallman and H.M. Oonk, The Use of 2D and 3D Displays for Shape-understanding versus Relative-position Tasks, *Human Factors*, Volume: 43, Number: 1, Page(s): 79-98, 2001.

[66] E. Kandogan, Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions, *The Procceddings of IEEE Information Visualization*, Hot Topics, Page(s): 4-8, 2000.

[67] D.A. Keim, Visual Database Exploration Techniques, *Procceedings of Tutorial KDD '97 International Conference Knowledge Discovery and Data Mining*, 1997.

[68] D.A. Keim and H.P. Kriegel, VisDB: Database Exploration using Multi-dimensional Visualization, *Computer Graphics & Applications*, Pages. 40-49, 1994.

[69] D.A. Keim and H.P. Kriegel, Visualization Techniques for Mining Large Databases: A Comparison, *The IEEE Transactions on Knowledge and Data Engineering*, Volume: 8, No. 6, Page(s). 923-936, 1996.

[70] D.A. Keim, H.P. Kriegel, and M. Ankerst, Recursive Pattern: A Technique for Visualizing Very Large Amounts of Data, *The Proceedings of the 6th conference on Visualization 1995*, Atlanta, GA, Page(s). 279-286, 1995.

[71] E. Kleiberg, H.van.de Wetering, and J.J. van Wijk, Botanical Visualization of Huge Hierarchies, *The Procceedings of the IEEE Symposium. on Information Visualization (InfoVis 2001)*, Page(s): 87–94, IEEE Computer Society Press, San Diego, California, 2001.

[72] J. Kort, Conference on Human Factors in Computing Systems 2005, (CHI 2005), *Workshop 6 Results*, April 2005. http://usage.nl/workshop.html.

[73] J. Kort and H.de Poot, Usage Analysis: Combining Logging and Qualitative Methods, *The Procceedings of the Conference on Human Factors in Computing Systems 2005 (CHI 2005),* Pages: 2121 – 2122, ACM, Portland, Oregon, USA, April 2005.

[74] K. Koffka, *Principles of Gestalt Psychology*, New York: Harcourt, Brace, 1935.

[75] J. Lamping, R. Rao, and P. Pirolli, A Focus plus Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies, *The Procceedings. SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, Page(s): 401–408. ACM Press, Denver, Colorado USA, 1995.

[76] T. Landauer, Research Methods in HCI. In M. G. Helander, editor, *Handbook of Human-Computer Interaction*, Chapter 42, Page(s): 905–928. North-Holland, 1988.

[77] R. Laurini and D. Thompson, Fundamentals of Spatial Information System, *Academic Press*, New York, 1995.

[78] J. LeBlanc, M.O. Ward, and N. Wittels, 'Exploring N-Dimensional Databases', *Proceedings of the 1st conference on Visualization 1990*, San Francisco, CA, Page(s): 230-239, 1990.

[79] H. Levkowitz, Color Icons: Merging Color and Texture Perception for Integrated Visualization of Multiple Parameters, *The Proceedings of the 2nd conference on Visualization 1991*, San Diego, CA, Page(s): 22-25, 1991.

[80] C. Lewis, P.G. Polson, C. Wharton and J. Rieman, Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces, *The Proc. Conference on Human Factors in Computing Systems (CHI 1990)*, Page(s): 235–242, ACM, Seattle, Washington, USA.

[81] C. Lewis and J. Rieman, *Task-Centered User Interface Design: A Practical Introduction, Shareware Book*, University of Colorado, Boulder. http://hcibib.org/tcuid/. (Cited on pages 96, 97 and 119.)

[82] D. Marr: Vision, *A computational Investigation into the Human Representation and Processing of Visual Information*, Freeman, San Francisco. 1982.

[83] D.J. Mayhew, *Principles and Guidelines in Software User Interface Design*, Prentice-Hall, 1991. ISBN 0137219296 (com, uk).

[84] G. Melancon and I. Herman, Circular Drawings of Rooted Trees, *Technical Report*: INS-R9817, CWI (Centre for Mathematics and Computer Science), Amsterdam, The Netherlands, 1998.

[85] D. Modjeska, V. Tzerpos, P. Faloutsos, and M. Faloutsos, BIVTECI: A bibliographic visualization tool. *The Proceedings of the 1996 Conference of the Centre of Advanced Studies on Collaborative Research*, Page(s): 28, 1996.

[86] T. Munzner, H3: Laying Out Large Directed Graphs in 3D Hyperbolic Space, *The Procceedings of IEEE Symposium on Information Visualization (InfoVis '97)*, Page(s): 2–10. IEEE Computer Society Press, Phoenix, Arizona, USA, 1997.

[87] Q.V. Nguyen , M.L. Huang, Space-optimized Tree: a Connection+Enclosure Approach for the Visualization of Large Hierarchies, *Information Visualization*, Volume: 2, Number: 1, Page(s): 3-15, March 2003.

[88] J. Nielsen, *Heuristic Evaluation*, http://www.useit.com/papers/heuristic/

[89] J. Nielsen, Enhancing the Exploratory Power of Usability Heuristics, *The Proc. Conference on Human Factors in Computing Systems (CHI 1994)*, Page(s): 152–158, ACM, Boston, Massachusetts, USA.

[90] J. Nielsen and R. L. Mack, *Usability inspection methods*, John Wiley & Sons, Inc., New York, NY, 1994.

[91] J. Nielsen and R. Molich, Heuristic Evaluation of User Interfaces, *The Procceddings of. Conference on Human Factors in Computing Systems (CHI 1990)*, Page(s): 249–256. ACM, Seattle, Washinton, USA, 1990.

[92] J. Nielsen and M. Tahir, *Homepage Usability: 50 Websites Deconstructed*, New Riders, 2001, ISBN 073571102X (com, uk), (Cited on page 95.).

[93] C. North, B. Shneiderman, and C. Plaisant, User controlled overviews of an image library: A case study of the Visible Human, *Procceedings of 1st ACM International Conference on Digital Libraries* , Page(s): 74-82, 1996.

[94] M.C.F.D. Oliveira and H. Levkowitz, *On-Line Resource on Visual Data Mining*, 2001, http://www.cs.uml.edu/~mcristin/vdm-resource.htm.

[95] R.M. Pickett, *Visual Analyses of Texture in the Detection and Recognition of Objects*, Picture Processing and Psycho-Pictorics, Lipkin B. S., Rosenfeld A. (eds.), Academic Press, New York, 1970.

[96] R.M. Pickett and G.G. Grinstein, Iconographic Displays for Visualizing Multi-dimensional Data, *The Procceddings of IEEE Conference,* 1988.

[97] C. Plaisant, The Challenge of Information Visualization Evaluation, *The Proceedings of the Working Conference on Advanced Visual Interfaces*, Page(s): 109-116, Gallipoli, Italy, May 25-28, 2004.

[98] C. Plaisant, B. Milash, A. Rose, S. Widoff and B. Shneiderman, LifeLines: Visualizing personal histories, *Proceedings of ACM CHI96 Conference: Human Factors in Computing Systems*, Page(s): 221-227, 1996.

[99] G. Polya, How to Solve It: A New Aspect of Mathematical Method, Princeton University Press, 1973

[100] D.J.de.S. Price, Network of Scientific Papers, Science, Volume: 149, Number: 3683, Page(s): 510-515, July 30, 1965

[101] E. Rasiel, *The Mckinsey Way*, McGraw-Hill Companies, 1998, ISBN-10:0070534489

[102] J. Raskin, *The Humane Interface*, Addison-Wesley Professional. ISBN 0201379376 (com, uk) . (Cited on pages 1and 96.).

[103] E.M. Reingold and J.S. Tilford, Tidier Drawing of Tree, *The IEEE Transcations on Software Engineering*, Volume: 7, Number: 2, Page(s): 223-228, 1981.

[104] J. Rekimoto and M. Green, The Information Cube: Using Transparency in 3d Information Visualization, *The Procceedings of 3rd Annual Workshop on Information Technologies & Systems (WITS '93),* Page(s): 125-132, 1993.

[105] G. Robertson , S.K. Card, and J.D. Mackinlay, The Cognitive Coprocesser for Interactive User Interfaces, *ACM Symposium on User Interface Software and Technology*, ACM Press, Page(s): 10-18, 1989.

[106] G. Robertson, R. Fernandez, D. Fisher, B. Lee, J. Stasko, Effectiveness of Animation in Trend Visualization, *IEEE Transactions on Visualization and Computer Graphics*, Volume:4, Issue:6, Page(s): 1325-1332, 2008

[107] G.G. Robertson, S.K. Card, and J.D. Mackinlay, Information Visualization Using 3D Interactive Animation, *Communication of the ACM*, Volume: 36, Number: 4, Page(s): 57-71, 1993.

[108] G.G. Robertson, J. D. Mackinlay, and S. K. Card, Cone Trees: Animated 3D Visualizations of Hierarchical Information, *The Proccedings of. Conference on Human Factors in Computing Systems (CHI 1991)*, Page(s): 189-194, 1991.

[109] G.G. Robertson and J.D. Mackinlay, The Document Lens, *The proceeding of 6th Annual ACM symposium on User Interface Software and Technology (UIST '93),* Page(s): 101–108. ACM Press, Atlanta, Georgia, USA, 1993.

[110] R.E. Rubin, *Foundations of Library and Information Science 2nd ed*, New York: Neal-Schuman, 2004.

[111] H. Schenker, *Five Graphic Music Analyses*, Dover Publications, New York, 1969

[112] A.J. Sellen, R. Murphy, and K.L. Shaw, How Knowledge Workers Use the Web, *The Procceedings of. Conference on Human Factors in Computing Systems (CHI 2002),* Page(s): 227–234, ACM, Minneapolis, Minnesota, USA, April 2002.

[113] B. Shneiderman, Tree Visualization with Treemaps: A 2D Space-Filling Approach, *ACM Transactions on Graphics*, Volume: 11, Number: 1, Page(s): 92-99, 1992.

[114] B. Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualization， *The Procceedings of IEEE Workshop Visual Languages*, Page(s): 336-343 IEEE Computer Society Press, Boulder, Colorado, USA, 1996.

[115] S.L. Smith and J.N. Mosier, *Design Guidelines for Designing User Interface Software*, The MITRE Corp., 1986, ISBN 9992080418.

[116] H.S. Smallman, M.St. John, H.M. Oonk and M.B. Cowen, Information Availability in 2D and 3D Displays, *IEEE Computer Graphics and Applications*. Volume: 21, Number: 5, 2001, Page(s): 51-57.

[117] R. Spence, L. Tweedie, H. Dawkes, and H. Su, Visualization for Functional Design, *The Procceedings of International Symposium on Information Visualization (InfoVis '95),* Atlanta, GA, Page(s): 4-10, 1995.

[118] R. Spencer, The Streamlined Cognitive Walkthrough Method, Working Around Social Constraints Encountered in a Software Development Company, *The Procceedings of Conference on Human Factors in Computing Systems (CHI 2000)*, Page(s): 353–359, ACM, The Hague, The Netherlands, 2000.

[119] R.R. Springmeyer, M.M. Blattner, and N.L. Max, A Characterization of the Scientific Data Analysis Process, *The Procceedings of IEEE Visualization*, pp 235-242, 1992.

[120] J.H. Siegel, E.J. Farrell, R.M. Goldwyn and H.P. Friedman, *The Surgical Implication of Physiologic Patterns in Myocardial Infarction Shock. Surgery*, 1972, pp: 126–141.

[121] S.T. Teoh and K.L. Ma, RINGS: A Technique for Visualizing Large Hierarchies, Lecture Notes In Computer Science, Volume: 25-28, *Revised Papers from the 10th International Symposium on Graph Drawing*, August 26 - 28, 2002

[122] E.R. Tufte, The Visual Display of Quantitative Information, *Graphics Press*, Cheshire, CT, 1983.

[123] E.R.A Valiati, M.S. Pimenta, and C.M.D.S Freitas, A Taxonomy of Tasks for Guiding the Evaluation of Multi-dimensional Visualizations, *The Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, Page(s): 1 – 6, 2006.

[124] J.Q. Walker II, A Node-Positioning Algorithm for General Trees, *Software-Practice and Experience*, Volume: 20, Number: 7, Page(s): 685–705. John Wiley and Sons, Inc, 1990.

[125] C. Ware, *Information Visualization: Perception for Design*, Mogan Kaufmann, 2004.

[126] M. Wattenberg, Arc Diagrams: Visualizing Structure in Strings, *The Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002)*, Page(s):110-116, October 28-29, 2002.

[127] C. Wharton, J. Rieman, C. Lewis, and P. Polson, *The Cognitive Walkthrough Method: A Practitioner's Guide*, Pages 105–140, John Wiley & Sons, Inc. New York, NY, USA.

[128] J.J.van. Wijk and R.van. Liere, Hyperslice. *The Procceedings of Visualization 1993*, San Jose, CA, Page(s): 119-125, 1993.

[129] 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, *The Proceedings of the IEEE Symposium on Information Visualization*, Page(s): 51–58, 1995.

[130] J.O. Wobbrock, J. Forlizzi, SE. Hudson, and B.A. Myers, WebThumb: interaction Techniques for Small-Screen Browsers, *The Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, Paris, France, Page(s): 205-208, 2002.

[131] J.O. Wobbrock, J. Forlizzi, SE. Hudson, and B.A. Myers, WebThumb: Interaction Techniques for Small-Screen Browsers, *The Proceedings of the 15th annual ACM symposium on User Interface Software and Technology*, Paris, France, Page(s): 205-208. 2002.

[132] P.C. Wong and R.D. Bergeron, 30 Years of Multi-dimensional Multivariate Visualization, Scientific Visualization Overviews, *Methodologies, and Techniques*, Los Alamitos, IEEE Computer Society Press, Page(s): 3-33, 1997.

[133] W. Wright, Information Animation Applications in the Capital Markets, *The Procceedings of International Symposium on Information Visualization*, Atlanta, GA, Page(s): 19-25, 1995.

[134] J. Yang , Matthew O. Ward , Elke A. Rundensteiner, InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures, *Proceedings of the IEEE Symposium on Information Visualization (InfoVis 2002)*, Page(s): 77, October 28-29, 2002

[135] Citation Analysis, Wikipedia, http://en.wikipedia.org/wiki/Citation_analysis

[136] CiteSeerX, http://citeseerx.ist.psu.edu/

[137] Computer Science, Wikipedia, http://en.wikipedia.org/wiki/Computer_science

[138] Eigenfactor Project, http://eigenfactor.org/

[139] EigenfactorTM Score, http://www.eigenfactor.org/methods.htm

[140] IEEE Xplore: http://ieeexplore.ieee.org/

[141] Hyperbolic Browser Demo, http://www.mtm.kuleuven.be/Research/Mentor-C/Prowat-demo/

[142] Inxight Star Studio: http://www.businessobjects.com/company/acquisitions/inxight.asp

[143] I Deal Solution: 3D Sphere-based Stock Market http://www.idealsolution.eu/

[144] InfoShape: High-level View of Multi-dimensional Information, Kang Zhang, Jie Hao. Provistional Patent, U.S. Patent Application No.: 61295452.

[145] JCR Citation Patterns, http://well-formed.eigenfactor.org/radial.html

[146] Journal Citation Reports on the Web 4.0, http://science.thomsonreuters.com/m/pdfs/mgr/jcr4_sem_0305.pdf

[147] Mapping the sphere, http://math.rice.edu/~polking/cartography/cart.pdf.

[148] Moritz Stefaner Visualization, http://moritz.stefaner.eu/

[149] OLIVE 1999: On-line Library of Information Visualization Environments. http://otal.umd.edu/Olive/

[150] Portfolio Map. http://www.smartmoney.com/

[151] Problem Solving, http://en.wikipedia.org/wiki/Problem_solving

[152] Stock Market Ticker Garden. http://infosthetics.com/archives/2006/04/ticker_garden_stock_market_visualization.html.

[153] Stock Market Planetarium. http://www.blackshoals.net/

[154] Spherical Helix Function: http://mathworld.wolfram.com/SphericalHelix.html

[155] TimeSketch, ECS Media, www.ecsmedia.com, 2002

[156] Z. Pizlo, 3D Shape-its Unique Place in Visual Perception, Cambridge, MA: MIT Press, 2008.

VITA

Jie Hao was born in Shijiazhuang, China in 1983. He received his Bachelor degree from the Department of Applied Mathematics, Beijing Jiaotong University, Beijing, China in 2005. After that, he started to pursue his doctoral degree in Computer Science of The University of Texas at Dallas in 2006. His research interests mainly focus on information visualization.