1-1-2010

Financial data visualization based on power-law degree distribution

Haibei Wu
Ryerson University

FINANCIAL DATA VISUALIZATION BASED ON POWER-LAW DEGREE DISTRIBUTION

by

HAIBEI WU
BEng, Tianjin Agricultural University, Tianjin, China Jun. 2006

A thesis
presented to Ryerson University
in partial fulfillment of the
requirement for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature: _____

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature: _____

BORROWER'S PAGE

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Name	Signature	Address	Date

Financial Data Visualization based on Power-Law Degree Distribution

Master of Applied Science 2010

Haibei Wu in the Program of Electrical and Computer Engineering

Ryerson University

Abstract

Problems in the field of data visualization are "how to define the layout algorithm to discover the hidden relationships in the financial data" and "how to design and develop a reliable system to implement the layout algorithm". Although there were many solutions to deal with the problems, the methods mainly focused on visualizing network topologies or social networks. This thesis develops a novel tree layout algorithm based on a power-law degree distribution and a robust flash web application system to investigate relationships in financial time-series data. Compared to previous tree layout algorithms, this novel layout algorithm is flexible, scalable, and easy to implement. Additionally, this thesis presents the design structure and important features in the novel graph visualization system. This system provides a number of unique characteristics. For example, if a new focus node is selected, the feature of *animating zooming with degree separations* will track an animating transition from one to another layout for users. At last, based on many experiments, the thesis demonstrates the profound applications and practices of this system. When utilizing the visualization system to display the data in world crude oil price, we discover that the price fluctuations of crude oil within Organization of the Petroleum Exporting Countries(OPEC) or non-OPEC group are correlated, but price fluctuations of crude oil between different groups are uncorrelated.

Acknowledgments

I would like to thank all the people who have supported me during the completion of this thesis. Their help has made this thesis possible and an enjoyable experience for me. I am very grateful to my supervisor, Dr. Xiao-Ping Zhang, for his guidance, encouragement, and support. Besides providing constructive discussions, his kind help is crucial for me to build my technical writing skills and the right attitude. I will definitely benefit from those research skills and the knowledge he shared with me.

I would also like to thank every member of my thesis defense committee. I appreciate their time, efforts, and contributions to this work.

Special thanks go to my colleagues in Communications and Signal Processing Applications Laboratory (CASPAL). It has always been a pleasure to discuss technical issues and exchange ideas with them.

I am also very grateful to the department of Computer Networks at Ryerson University for the financial support and the comfortable research environment.

Lastly, I would like to show my gratitude to my family for their caring and support. I dedicate this thesis to my parents and my girlfriend Shuyi Feng.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the information increase in the cyber space, there are growing needs in the field of data visualization which is close related to graphic design and information representation. Herman et al.[1] pointed that a simple way to determine the applicability of graph visualization is to consider the following question: Is there an inherent relation among the data elements to be visualized? If the answer to the question is "no", then data elements are unstructured, and the goal of the information visualization system is to help users discover relationships among data through visual meanings. If the answer to the question is "yes" then the data can be represented by nodes of a graph, with edges representing the relations. This thesis assumes that there is an inherent relation within a certain type of time-series data.

In particular, based on many layout results, the method in data visualization generates a lot meaningful results in the financial data. Through analyzing the visualized result in the novel graph visualization system, we can easily extract and explore the structure among the financial data. By doing so, we can verify our assumption. For example, when using the hierarchy stratification algorithm to visualize price correlations between countries in the world crude oil market, one can employ the visualized graph to verify an assumption that the price fluctuations of crude oil within Organization of the Petroleum Exporting Countries(OPEC) or non-OPEC group are correlated, but price fluctuations of crude oil between different groups are uncorrelated or not that strongly correlated.

1.1 Motivation and Objectives

Graph visualization is widely used to visualize network topologies and social networks. There are few visualization technologies which have been applied in the financial industries. The structure of the graph is inherent in the network topologies. If the original data has a clear structure, the goal of this layout algorithm is to find a best layout view to represent the structure. When using visualization technologies to explore the structure in financial time-series data, one must define a type of relation which estimates if there is a connection or an edge between two nodes in a graph.

In most popular visualization technologies, there is a typical hierarchy layout method, which builds a graph hierarchy by stratifying nodes into different levels so that central and representative nodes in the graph were emphasized. This technology was utilized to visualize price return correlations between stocks in the Standard & Poor's 500 (S&P 500), where two stocks are connected by an edge when their returns correlations are above a preset threshold. This algorithm contains some constraints and limitations, which requires vertex degrees in a graph to be distributed according to a power-law. When using this algorithm to visualize other financial data, such as crude oil price data and agriculture fundamental data, it was discovered that the node degree distribution in a graph was not perfectly fitted in the power-law distribution. In this case, the visualization system based on this technology did not generate a meaningful result. In other words, if the degree distribution of nodes is not fitted in the power-law distribution, the method that originates from the Riemann-Zeta function [2] in calculating depth of the hierarchy tree is not adaptive. Therefore, a new formula must be created as an alternative way to evaluate the depth of the hierarchy layout. The main objectives of this thesis deal with the limitations and constraints in the above method and develop a novel hierarchy layout algorithm for exposing the structure of graphs where the degree distribution is not according to the power-law distribution.

Moreover, we also found that most visualization systems were concentrating on implementing specific visualization algorithms. These systems did not provide enough interactive actions to users, so that the users are difficult to affect the graph result by their operations.

In order to improve the user experience, a robust multi-functional visualization system based on the web application was created. Our system does not only implement our specific hierarchy algorithm, but it also provides more interactive features to users in order to let users see a variety of perspectives in the graph visualization results.

1.2 Review of Previous Works

This thesis focuses on the layout of undirected graphs to visualize financial time-series data. In early successful graph layout algorithms, force-directed algorithm were typically applied to relatively small and sparse graphs. Eades [3] pioneered this approach and developed the earliest spring-embedded model for graph drawing. The method modeled each node as a positive charge and each edge as a spring. The charges (nodes) repelled each other while the springs (edges) attracted the charges. For each node, attractive and repulsive forces were calculated with respect to the other nodes in the network. The positions of the nodes were updated according to the forces on each node, until the network settled into a minimum energy state.

Kamada-Kawai algorithm [4] is a variant of Eades' Spring model, which does not place charges on the nodes. Instead, each node is connected together by springs, forming a complete graph. This approach generated aesthetically pleasing results. However, this situation requires calculating every pair of nodes in the network, which posed serious complexity issues for large graphs with more than a hundred nodes.

Fruchterman and Reingold [5] developed another variant of Eades' spring model, and they simplified the force calculation by using attractive forces only between neighboring nodes, while repulsive forces were still calculated between all nodes in the graph. Gautam Kumar and Michael Garland [6] also calculated forces between two nodes using the Fruchterman-Reingold model.

A major drawback of the force-directed approach was the complexity of calculating the forces between all pairs of nodes. An alternative way was to use a multi-level or hierarchical approach to graph layout. The original graph was abstracted into a hierarchy of simpler

graphs, either by clustering similar nodes, or by selecting a small set of influential nodes. The process was repeated at each level of the hierarchy until a layout had been found in an original graph. For instance, Walshaw [7] designed a hierarchical approach based on pair-wise clustering of nodes and a force-directed layout. The main advantage of this approach provides a global solution for solving the layout problem, which was first made on a simplified graph requiring less computation subsequent level of detail so that, overall, less time was spent in optimizing the original graph. The success of this approach relies on having a suitable strategy for simplifying the graph at each level of abstraction. Actually, an efficient strategy is to abstract graphs and capture the main features.

In order to avoid repeated coarsening in the hierarchy layout approach, Chan et al. [8] developed a novel hierarchical layout algorithm called OutDegree Layout(ODL), which was designed to exploit the power-law distribution that was commonly observed in many kinds of practical networks. Their approaches were based on the idea that nodes with a high outdegree played a major role in determining the final structure of a graph, while nodes with a low outdegree have a less effect. Their aims were to decrease the time complexity of a graph layout algorithm and generate an aesthetically pleasing layout. In addition, Kumar et al. [6] developed their interactive hierarchy layout algorithm based on the power-law distribution. They constructed a graph hierarchy by stratifying nodes into different levels, and emphasized the central and representative nodes in their visualized graph.

Related to previous works, we can find that there were many kinds of graph layout methods which were utilized in the data visualization or information visualization. In contrast to prior works, which Chan et al. [8] and Kumar et al. [6] explored their algorithms based on degree power-law distribution, we find that there are some limitations and constraints in the two algorithms. The ODL algorithm visualized topologies in the network fields where the relation between two nodes have been determined by topologies. Many observations have been verified that the node degree of the inter-domain network topologies were distributed according to the power-law. The hierarchy stratification algorithm developed by Kumar et al. verified that their similar layout algorithm did work well in the financial time-series

data. The quality of their hierarchy layout algorithm was still very data-dependent. In other words, the hierarchy imposed on the graph may not be as meaningful in graphs that did not match the power-law distribution of edges.

Based on the above discussion, we learned the advantages and disadvantages in the previous graph layout algorithms. In this thesis, we concentrate on solving the problems in the previous layout algorithm and improving the performance in our current new application.

1.3 Summary of Contributions

The main contributions of this thesis are to improve the hierarchy layout algorithm based on a power-law degree distribution and develop a novel interactive and dynamic visualization system in a flash web application system. We explore a new flexible and scalable hierarchy layout algorithm to visualize the financial time-series data. The novel layout method is on the top of the previous hierarchy layout algorithm with the power-law distribution. For comparison, we will mainly point out where our approach differs from the techniques of Kumar et al.. Additionally, most recent research working on the graph layout algorithm mainly concentrated on implementing the algorithm into the network topologies in telecommunications or social networks. And most graph visualization systems only generated a static graph or tree result for implementing a single or specific layout algorithm. For improving and solving the existing problems in this graph layout algorithm, we developed an impressive visualization system with multiple new features and functions. This system is a very useful platform to demonstrate our hierarchy stratification algorithm. By using our multi-functional visualization system with a novel hierarchy layout algorithm, we experimented many databases in financial time-series data and generated meaningful and understandable layout results. Figure 1.1 can demonstrate our summary of contributions in this thesis.

In this novel hierarchy algorithm, we add a threshold selection method for constructing a graph, where two nodes are connected by an edge when their value of correlation coefficient is above a selected threshold. In contrast to Kumar et al. [6] layout algorithm, there was no methodology in selecting the threshold value and the reason why one selected a threshold.

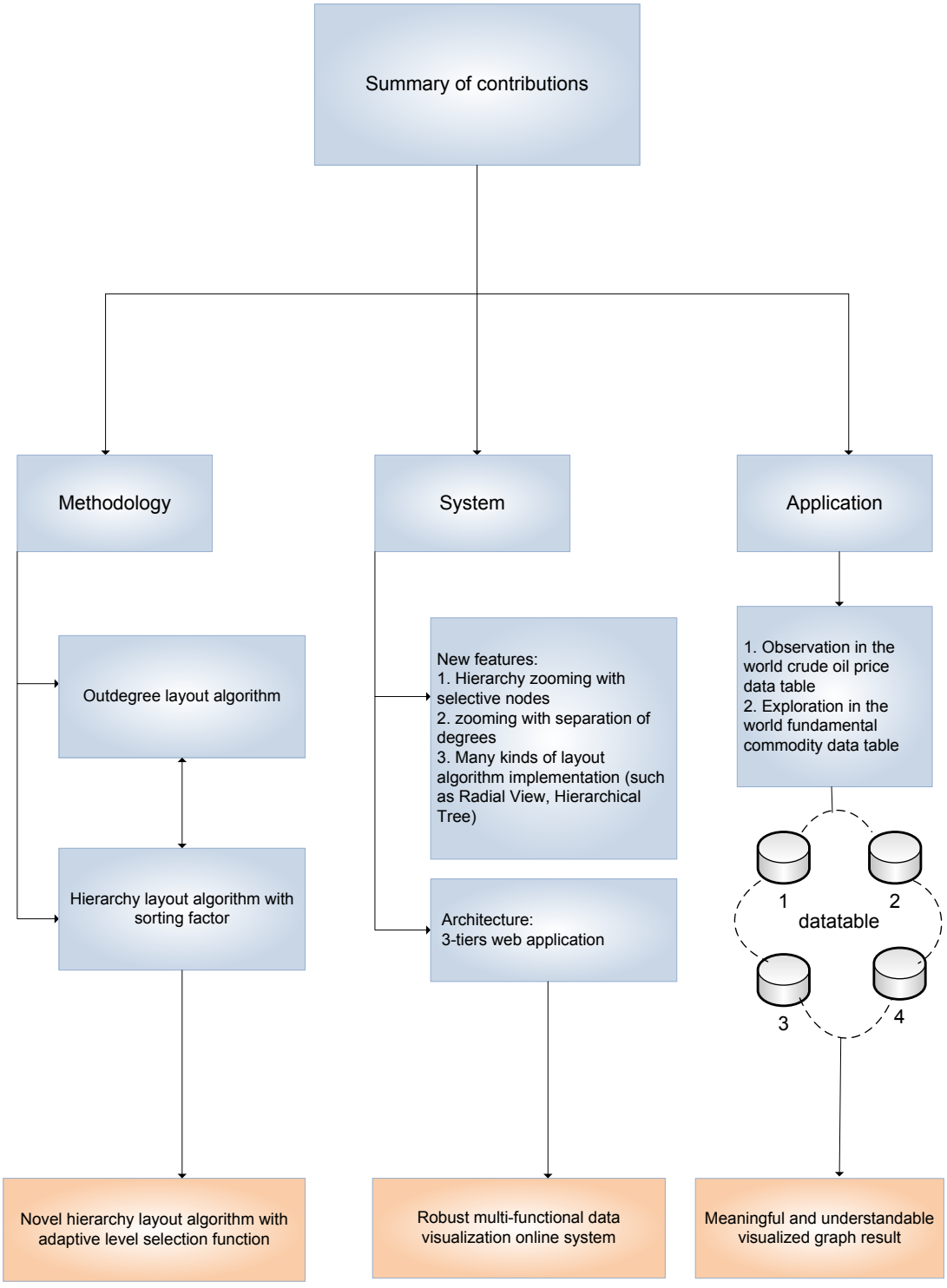


Figure 1.1: Summary of contribution

This thesis gives an exhaustive analysis on the threshold selection. The main target of this threshold selection method is to construct a degree distribution according to the power-law distribution. Besides, compared to Chan et al. [8] and Kumar et al. [6] hierarchy layout algorithms, our novel algorithm contains different processing methods in a construction of the hierarchy layout, including ranking nodes, stratifying graphs and computing families. The improved algorithm has good performance in the visualization of financial time-series data, and gains a meaningful result in the underlying graph structure.

Another main contribution in this thesis is the design and construction of a novel interactive visualization system to demonstrate the hierarchy layout algorithm. The dynamic visualization system provides users different kinds of visualized graphs and helps them understand the financial time-series data in diverse perspectives. For example, the graph system visualizes the hierarchy result using many kinds of popular layout algorithms, such as concentric radial, single cycle circle, hyperbolic, hierarchical tree etc. In particular, the visualization system includes some unique features. For example, a selective node can zoom in the center of a graph when the user selects a new focus of the graph, and meanwhile this transition from one layout view to another is displayed in our animated visualization system. Moreover, the graph layout result in our system can be displayed in different types of graph views, and smoothly changed in different levels with aesthetically pleasing effects. What is worth mentioning, our graph system is developed on a typical 3-tier architecture based on the Browser/Server (B/S) application. Similarly, our graph visualization system is an online tool and provides users with dynamic data retrieval from the database.

From another perspective, this thesis also promotes a novel layout algorithm which visualizes the relationships in the financial data to users. Chan et al. [8] developed the hierarchical layout algorithm for visualizing large network topologies; Kumar et al. created the hierarchical layout algorithm for verifying price fluctuations of stocks in the same industries are strongly correlated. Those applications strongly testified that the hierarchical layout algorithm based on the power-law distribution was reliable and dependable. In this thesis, in order to solidify this theory, the novel layout algorithm based on the power-law

distribution will be applied to a wide variety of data and gain meaningful layout results. For example, when the hierarchy layout algorithm is applied to the world crude oil price data, the layout results verify an assumption: The price fluctuations of crude oil within an OPEC or non-OPEC group are correlated, but price fluctuations of crude oil in different groups are uncorrelated or not as strongly correlated; when the hierarchy layout algorithm is applied to agricultural commodity data, the layout result also validates an assumption: The production fluctuations of agricultural commodity in the same continent are correlated, but production fluctuations of agricultural commodity in different continents are uncorrelated or not as strongly correlated.

1.4 Outline of Thesis

This thesis is organized as follows. Chapter 2 will introduce the basic concepts of graph visualization algorithms, and also point out what the limitations and constraints in the previous algorithm are; Chapter 3 will focus on describing our hierarchy layout algorithm; Chapter 4 will discuss the unique features in our visualization system, implementation of our layout algorithm in our visualization system, and the system structure of our visualization system; Chapter 5 will analyze the result based on our visualization system, and verify that our graph result is meaningful and helpful to users. Chapter 6 concludes our work and expect the future work.

The structure of the thesis is shown in Figure 1.2:

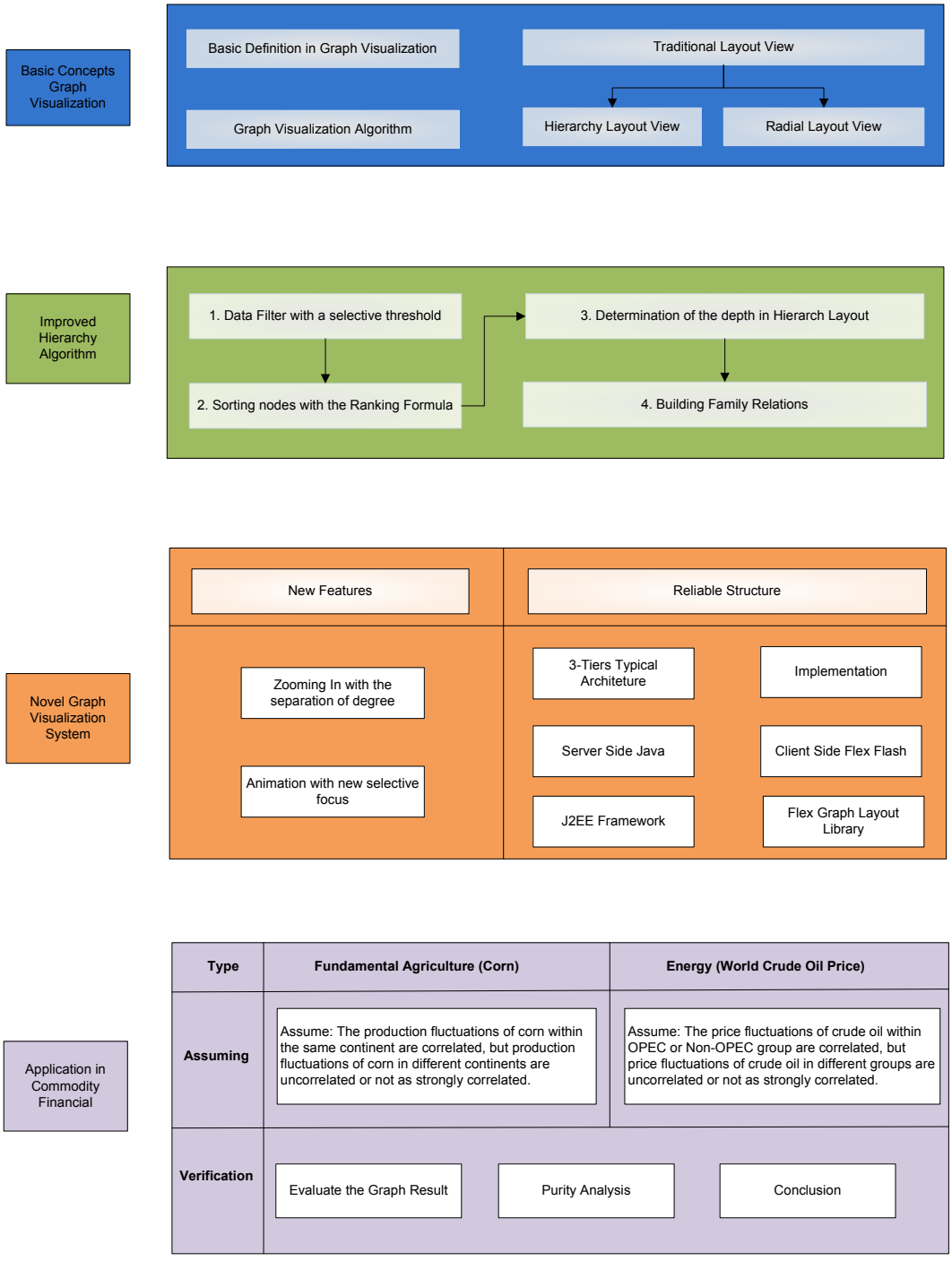


Figure 1.2: Thesis outline

Chapter 2

Preliminaries

In order to smoothly understand our novel algorithm and our graph visualization system, we discuss some basic concepts in graph visualization, including visualization technologies based on power-law distribution and traditional graph layout algorithms. Moveover, we also emphasize the constraints and limitations in the previous graph layout algorithms and technologies based on the power-law distribution.

2.1 Graph Visualization

The visualization of complex conceptual structures is a key component of support tools for many applications in science and engineering. A graph is an abstract structure that is used to model information. Graphs are used to represent information that can be modeled as objects and connections between those objects. Hence, many information visualization systems require graphs to be drawn so that they are easy to read and understand.

2.1.1 Basic Concepts for Graph Drawing

Graph drawing addresses the problem of constructing geometric representations of graphs, networks, and related combinatorial structures. Geometric representations of graphs have been investigated by mathematicians for centuries, for visualization and intuition, as well as for the pure beauty of the interplay between graph theory and geometry. In the 1960s, computer scientists began to use graph drawings as diagrams to assist with the understanding

of software. Knuth's [9] 1963 paper on drawing flowcharts was perhaps the first paper to present an algorithm for drawing a graph for visualization purposes.

Today, many applications are found in the area of graph drawing. Examples include software engineering, databases, information systems, real-time systems, decision support systems, artificial intelligence, and logic programming. Also, many more applications can be found in other science and engineering disciplines, such as medical science, biology, chemistry, civil engineering, and cartography. Because of the combinatorial and geometric nature of the problems investigated, and the wide range of the application domains, research in graph drawing has been conducted within several diverse areas, including discrete mathematics and algorithmic and human-computer interaction. In addition, various graphic standards are used for drawing graphs. Usually, vertices are represented by symbols such as points or boxes, and edges are represented by connection between the associated vertices. However, the graphic standards may be different depending upon the application. For example, mathematicians seem to prefer straight-line drawings, where edges are straight-line segments, while circuit and database designers tend to use orthogonal drawings, where edges consist of horizontal and vertical segments. Therefore, there is no uniform criterion or formula to judge if a drawing of a graph is useful or helpful. The usefulness of graph drawing depends on its readability, that is, the capability of conveying the meaning of the graph quickly and clearly. Readability issues can be expressed by means of aesthetic criteria, such as the minimization of crossings between edges, and the display of symmetries.

When drawing a graph, we would like to take into account a variety of aesthetic criteria. For instance, planarity and the display of symmetries are often highly desirable in visualization applications. In general, in order to improve the readability of a drawing, it is important to keep the number of bends and crossings low. Also, to avoid wasting space on a page or a computer screen, it is important to keep the area of the drawing small, subject to resolution rules.

Relational structures consist of a set of entities and relationships between those entities. Such structures are usually modeled as *graphs*: the entities are *vertices*, and the relationships

Symbol	Definition
G	An undirected graph.
V	A Collection of all nodes in an undirected graph.
N	Number of nodes in a graph.
E	Number of edges in a graph.
D	Depth in a graph
d_v	Outdegree of node v
v,u	Any node in an undirected graph.

Table 2.1: Definitions and symbols

are *edges*. A *graph* $G = (V, E)$ consists of a finite set V of *vertices* and a finite multi-set E of *edges*, that is, unordered pairs (u, v) of vertices. The *neighbors* of v are its adjacent vertices. The *degree* of v is the number of its neighbors. In my thesis, the *vertices* of a graph are sometimes called nodes; *edges* are sometimes called links, arcs or connections. Graphs include *directed* graphs and *undirected* graphs. In this thesis, we use an *undirected* graph to visualize the financial time-series data. Table 2.1 lists the symbols and definitions used in graphs.

2.1.2 Traditional Layout

In this section, we will simply review existing layout techniques in graph drawing, and mainly focus on some graph layout methods which have been used in our graph visualization system. A classical tree layout positions children nodes "below" their common ancestor. The algorithm by Reingold and Tilford [10], [11] is probably the best known layout technique in the tree layout category (see Figure 2.1). It can be adapted to produce top-down, as well as down-top, left-right, right-left tree layouts, and can also be set to output grid-like positioning. Hierarchy-tree(H-tree) layouts are also classical representations for binary trees which only perform well on balanced trees.

The radial positioning by Eades [9] places nodes on concentric circles according to their depth in the tree. A subtree is then laid out over a sector of the circle and the algorithm ensures that two adjacent sectors do not overlap. The cone tree algorithm can be used to

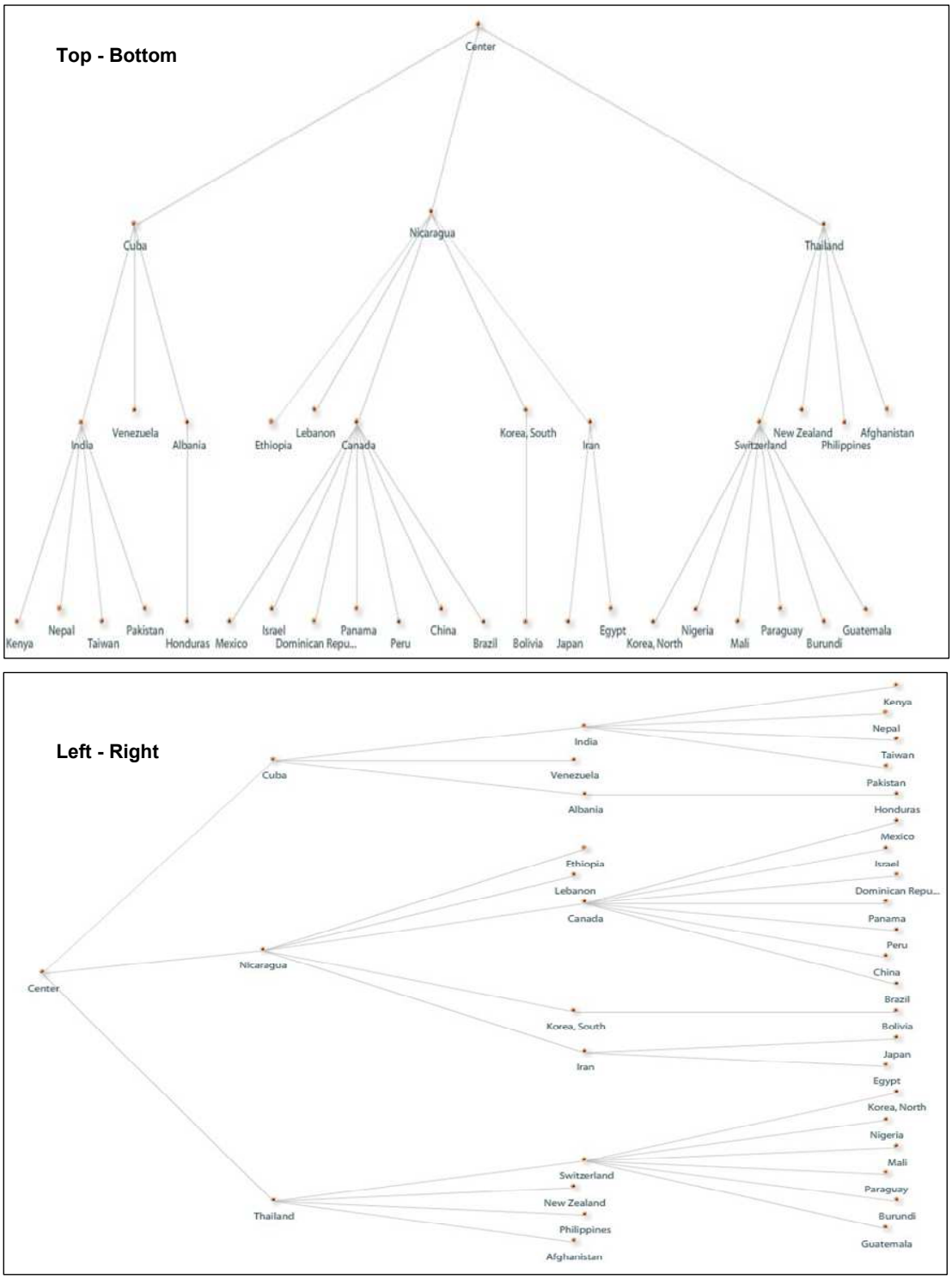


Figure 2.1: Hierarchy view

obtain a "balloon view" of the tree by projecting it onto the plane, where sibling subtrees are included in circles attached to the parent node.

The hyperbolic layout of graphs is one of the new forms of graph layout which has been developed with graph visualization and interaction in mind. Lamping et al. [12] first developed web content viewers based on these techniques. Hyperbolic tree can be implemented in either 2D or 3D.

The radial view, the H-tree and the hyperbolic tree are used to generate Non-Planar (NP) graph and avoid crossings in order to give users a more pleasing graph layout. In this thesis, we will mainly employ the hierarchical tree and radial tree to visualize the financial data. In the coming subsection, we will give more details about two layout algorithms.

Radial Tree

The radial view is one of the most popular visualization methods. The radial view can be represented by four main types: concentric radial tree, parent center radial tree, hyperbolic radial tree and single cycle circle radial tree. Figure 2.2 shows the four types of radial trees and demonstrates the properties and attributes in different radial trees. The concentric radial tree is to arrange nodes in a circular layout and position the root-node at the center of the graph and child-nodes in a circular fashion around the root. Sub-trees formed by the branching of the child-node are located radially around the child-node. Compared to the concentric radial tree, the parent center radial tree emphasizes the center nodes and highlight nodes in the center position. The single cycle circle radial tree simply assigns nodes in a cycle circle. The method of creating a hyperbolic radial tree is placing a node far enough from its parent such that it gives the node almost the same amount of space as its parent for laying out its own children.

In this thesis, the radial tree has the following properties:

- It requires 2D graphics display.

  In our radial tree, we use the 2D graphics display to visualize the economic market data. The 2D graphics display is a simple and clear visualization method, although

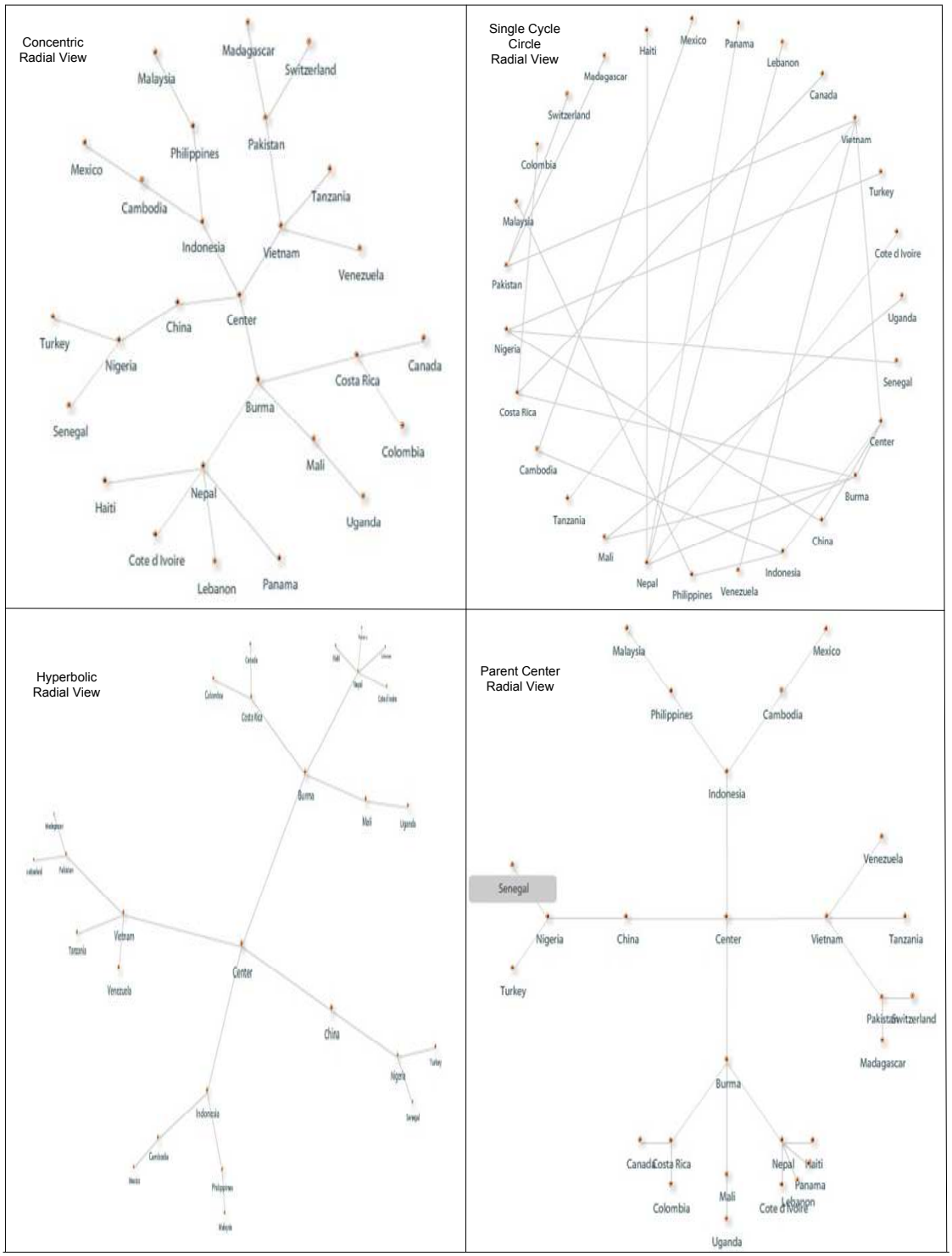


Figure 2.2: Four main types of radial trees

3D graphics display is becoming more and more popular. In many cases, 3D graphic structures are used for aesthetics. To gain a clear radial tree graph, we choose 2D radial tree to visualize the financial time-series data.

- It cannot visualize node content.

  The number of the nodes and the size of the screen are still bottlenecks in data display and graph layout. Therefore, in order to display the entire structure of the tree, the radial tree omits the content of the tree.

- It is suitable for an overview of a large graph

  Because the node content display is disabled, the radial tree focuses on showing the overview of a large graph. Although hundreds or thousands of nodes need to be displayed, the radial tree can deal with it properly. This overview allows clients to see the entire picture.

Hierarchical Tree

The hierarchical tree is one kind of tree structure in a graphic form. In the real world, the hierarchy structure is widely used. In particular, it is usually used to display the membership structure of a company. For example, Figure 2.3 shows a hierarchical structure of a company. By studying this figure, we can clearly see the organized structure in this company.

The hierarchical view of the tree is based on the well-known algorithm of Reingold and Tilford [10] revisited by Walker [11]. They developed a simple, fast, and completely predictable layout algorithm. There are various variants in the hierarchical layout, including grid-based, left-right and top-down, etc. All these variations were mathematically identical and implementers were tempted to arbitrarily include just one variant only. This would be a mistake: one should recognize that the way of looking at trees may depend on the application. For example, the top-down hierarchical view is the most widespread way of looking at family trees, whereas biological evolution schemas often use a left-to-right grid. The conclusion is that we give users the choice and let them choose their preferred views.

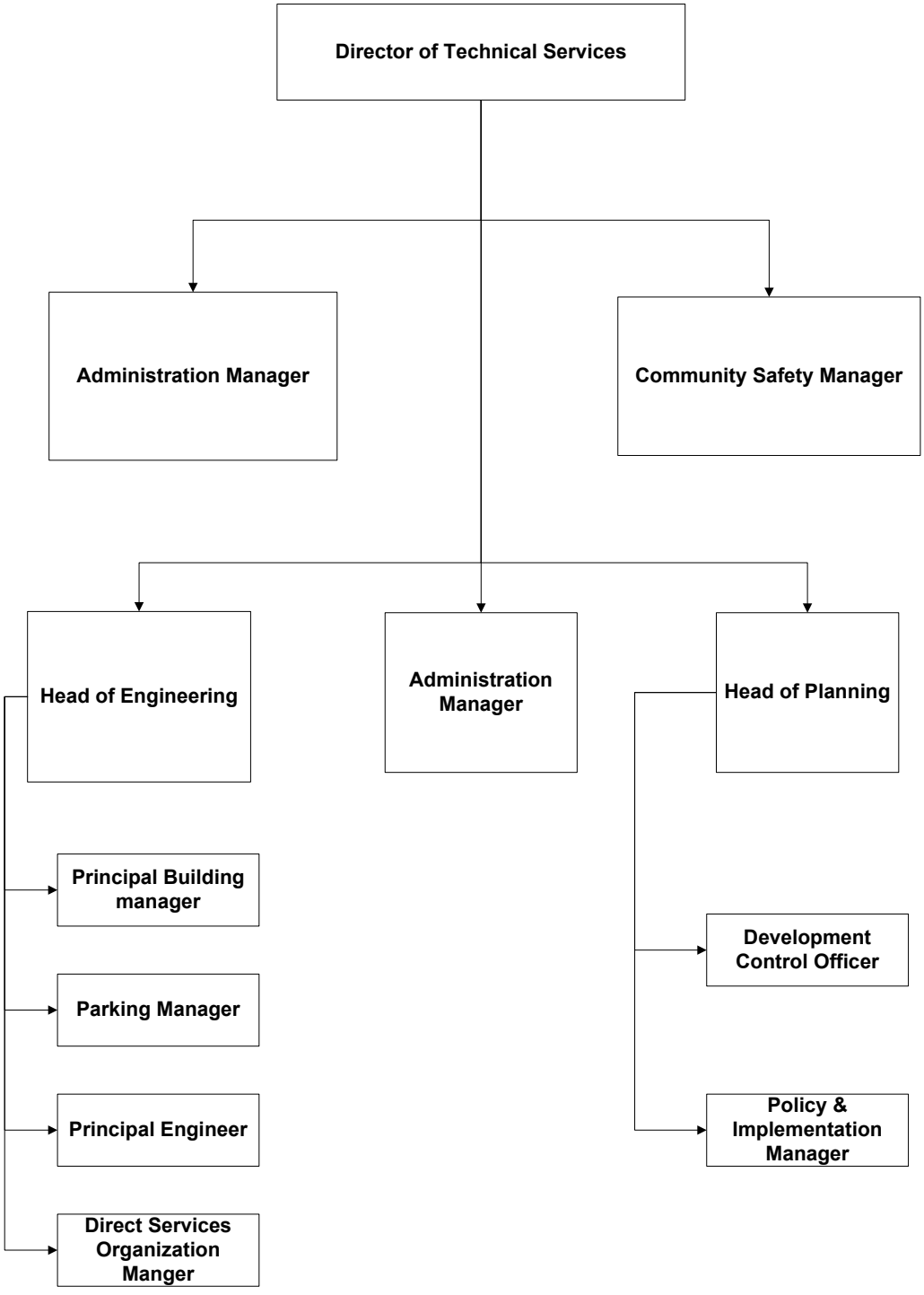


Figure 2.3: The example of hierarchy tree

Power-Law definition
Power-Law 1 (rank exponent) The outdegree, d_v, of a node v, is proportional to the rank of the node, r_v, to the power of a constant, R: $$d_v \propto {r_v}^R$$
Power-Law 2 (outdegree exponent) The frequency, f_d, of an outdegree, d, is proportional to the outdegree to the power of a constant, K: $$f_d \propto d^K$$
Power-Law 3 (eigen exponent) The eigenvalues, λ_i, of a graph are proportional to the order, i, to the power of a constant, ε: $$\lambda_i \propto i^{\varepsilon}$$

Table 2.2: Power-Law definitions

2.1.3 Graph Technologies Based On Power-Law Distribution

The power-law distribution is just one of many probability distributions, but it is considered a valuable tool to assess uncertain issues that the normal distribution cannot handle. Unlike the normal distribution, the power-law distribution does not have a well defined mean and variance. A power-law distribution is characterized by a large number of small observations and a small number of very large observations. In recent years, the power-law distribution has been explored and utilized in many fields, including general software systems [13], exact analysis of network topologies and traffic [14] [15], and social phenomena [16] etc.

Firstly, Faloutsos et al. [17] discovered some simple power-laws of the Internet topologies and defined three types of different power-law instances(see Table 2.2). In addition, they used power-laws and their approximations to estimate useful parameters of the Internet, such as the average number of neighbours within h hops. Their observations based on power-laws provided a novel perspective of the structure of the Internet.

Based on the Power-Law 2, Chan et al. [8] presented a novel graph layout algorithm

called ODL, which is designed to exploit the power-law distribution that is commonly observed for the outdegree of nodes in many kinds of practical networks. Their power-law approach described that nodes with a high outdegree play an important role in determining the final structure of the graph, while nodes with a low outdegree have far less impact. Their targets are to reduce the complexity of graph layout, and generate an aesthetically pleasing layout. Their hierarchy approach was not limited to a certain class of networks. They also demonstrated that their algorithm can generate useful and aesthetically pleasing layouts for a wide variety of networks, including both regular and power-law topologies in networks. In particular, their algorithm achieved performance improvements over existing layout techniques. However, this algorithm did not provide a method for determining the layers of a graph. And power-law rules were only applied to analyze the topologies in an network field.

With the development in technologies of graph visualization, power-law distribution was observed in the time-series stock datasets. Kumar et al. [6] developed a hierarchy layout algorithm, which constructed a graph hierarchy by stratifying nodes into different levels so that central and representative nodes in the graph were emphasized. Their stratified hierarchy exposed the underlying structure of non-planar graphs because vertex degrees of such graphs were distributed according to a power-law. As an example application, they utilized their tool to visualize price return correlations between stocks in the S&P 500. Compared to the method of utilizing power-law rules in the network topology visualization, they constructed a graph based on price return correlations, where two stocks were connected by an edge when their returns had power-law degree distributions [18] [19]. In addition, they built their visualization tool to achieve a much more aesthetically pleasing graph when viewed level by level. Several other graph visualization tools [20], built hierarchies based on graph coarsening with the goal of preserving the structure of the complex graph. But those visualization tools produced a lot of complex graphs which were confusing and unclear. Therefore, their hierarchy was instead based on edge distribution, and was built to reflect the underlying structure of complex graphs. Although their visualization tool was very useful for most data, the quality of the hierarchy created was still very data-dependent. The hierarchy

imposed on the graph may not be as meaningful for graphs which do not perfectly match a power-law distribution of edges.

Chapter 3

Hierarchy Layout Based On Degree Distribution

The power-law rules are commonly found in many practical networks such as the Internet [17] and the WWW [21], where a few visualization technologies were applied to visualize network topologies in a graph structure. The hierarchy layout algorithm based on power-law degree distribution was verified to be a novel visualization technology in analyzing stock data in the S&P 500. Based on the power-law distribution, Kumar et al. [6] developed their hierarchy layout algorithm which proved that most stocks' tree siblings were in the same industry. In fact, the evidence proving this algorithm was that the degree distribution in the nodes of a graph was perfectly matched to power-law distribution. However, if the degree distribution was not followed with a power-law distribution(Zeta distribution), they were unable to prove that their algorithm still had a good performance and meaningful graph layout result. In order to solve the problem, we find a solution to visualize the graph where the degree distribution is not perfectly distributed according to a power-law. Our hierarchy layout algorithm is designed and developed based on degree distribution. The core idea of this algorithm is that we place sorted nodes in different levels so that each level has an equal total number of degrees.

This chapter will mainly concentrate on explaining how to construct the hierarchy layout algorithm and include two parts: selecting a rational threshold and constructing a novel hierarchy layout. The former will focus on demonstrating a selective threshold method in

order to generate a power-law correlation metric. The latter will illustrate each important step in the algorithm and specifically emphasize the improvements compared to the previous algorithm.

3.1 Data Filtering Based On Correlation

In the data mining field, data filtering is always a key step in data pre-processing. In this thesis, the goal of data filtering is to make up our raw data so that it can help build a power-law degree distribution. Data Processing is always the first step when the work is related to data mining. One must have a comfortable data source or dataset whose structure is in accord with the chosen data mining technologies. Additionally, data processing covers a wide area which contains data filtering, data dimension reduction, data normalization, etc. This thesis concentrates on a data filter technology which is to ensures node degree distribution in the visualized graph is according to a power law, and to improve the application performance to some degree. In the following subsections, we focus on demonstrating our selective threshold method for data filtering.

3.1.1 Similarity Measures

Unlike other types of data structure such as social networks and network topologies, there is no directly connected relations in the financial data. In other words, the financial data are originally independent without any conditions. In this thesis, we give a condition (see Table 3.1) for determining whether there is a relation between each two time-series data in our financial data table. This condition is based on similarity measures between two data sequences which have the same length. The methods in similarity measures were discussed by Chen et al. [22] in the survey of data mining, where they reviewed basic similarity measures, and mainly compared Euclidean distance [23] and the correlation [24]. The Euclidean distance is only meaningful for measuring the distance between two vectors with the same dimension. In contrast to the method based on Euclidean distance, there are more benefits and advantages in the correlation methods. In some cases, the correlation function is

The condition in justifying relations between two time-series data
If the correlation value between two time-series data is larger than zero, we can determine whether there is a connection between them. $$\|r_{i,j}\| > 0$$ where $r_{i,j}$ represents the correlation value between data-series i and j.

Table 3.1: The condition in data filtering

called normalized Euclidean distance. Considering the advantages in correlation methods, we choose the Pearson correlation [25] as the similarity measure method.

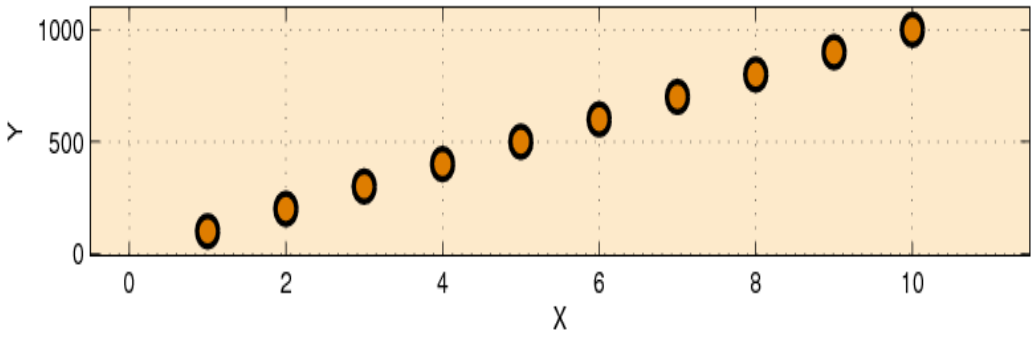
Let X and Y be independent random variables; each has multiple samples (X:x_i, Y: y_j). Pearson's correlation factor between x_i and y_j, r_{xy}, is defined as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{(n-1)S_x S_y} \tag{3.1}$$

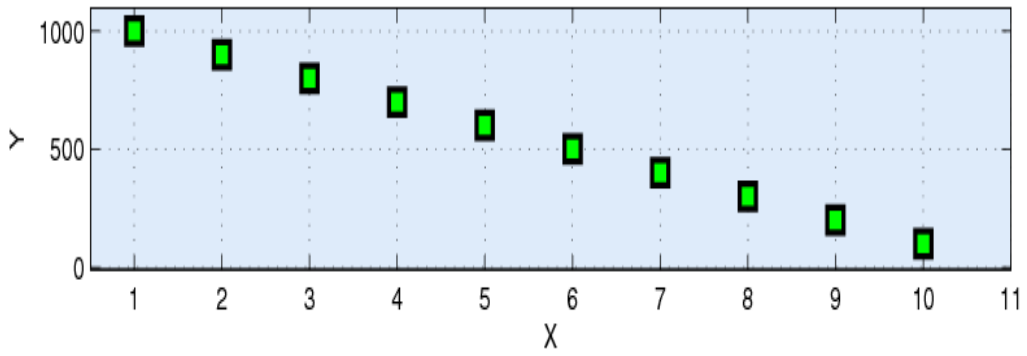
where $\overline{x}$ and $\overline{y}$ are the sample means of X and Y, S_x and S_y are the sample standard deviations of X and Y. The mean ($\overline{x}$) and sample standard deviations (S_x) of X , is computed as below:

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_i, S_x = \sqrt{\frac{1}{n-1}\sum_{k=1}^{n}(x_i - \overline{x})} \tag{3.2}$$

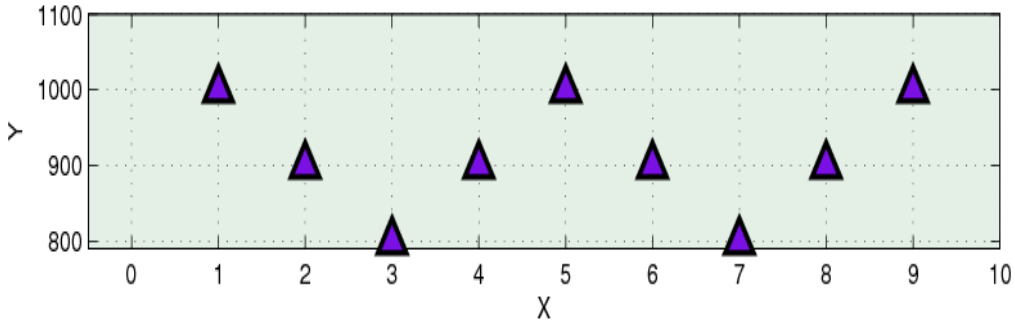
The value range of Pearson's correlation coefficient r_{xy} is $[-1, +1]$ which absolute value cannot exceed 1. The Pearson correlation coefficient equals +1 in the case of a perfect positive linear relationship between two variables. Instead, -1 represents the case of a perfect negative linear relationship, and some value between -1 and 1 occurs in all other cases [26]. In other words, the value of Pearson's correlation coefficient indicates the direction of association between X and Y (X,Y: the dependent random variable). If Y tends to increase when X increases, Pearson's correlation coefficient is positive. If Y tends to decrease when X increases, the Pearson's correlation coefficient is negative. Pearson's correlation of zero indicates that there is no tendency for Y to either increase or decrease when X increases (see Figure 3.1).



Pearson's Correlation Coefficient = 1



Pearson's Correlation Coefficient = -1



Pearson's Correlation Coefficient = 0

Figure 3.1: Pearson's correlation examples (value = -1, 0, +1)

3.1.2 Selective Threshold Method

According to the above definitions, we calculate a correlation matrix r to record the correlation value between each two data series. Assuming that the number of data series in our data table equals n, the matrix r is a n-n square matrix. In this case, each row or column in this matrix can represent a node in our graph and each two nodes have a correlation value $r_{i,j}$ in the correlation matrix r. If we predefine: "when the correlation value between two nodes is larger than zero, there will be an edge between them in the graph", the number of total degree for each nodes is always $n-1$. Because our hierarchy layout algorithm is based on the power-law degree distribution, we have to develop a method to preset a threshold in correlation value. The method concentrates on automatically selecting a threshold correlation value in order to construct a node degree distribution following a power-law. Our method can be illustrated within several steps:

1. Calculating global average correlation coefficient

   Global average correlation coefficient is an average value in the correlation matrix. The formula is shown below:

   $$\bar{r} = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |r_{i,j}|}{n^2} \tag{3.3}$$

   where $r_{i,j}$ is a value in the ith row and jth column of a correlation matrix r; n is the number of total nodes. $\bar{r}$ is the average correlation value in a global correlation matrix. The global average correlation coefficient is a threshold which can determine if there is a relation between each pair of nodes. (Each column or row in a correlation matrix can be a node)

2. Resetting the correlation value

   According to the above threshold, our data filtering algorithm will iterate through each values in this correlation matrix. If the correlation value between two nodes is smaller than the threshold, we reset the correlation value to zero, which represents that there is

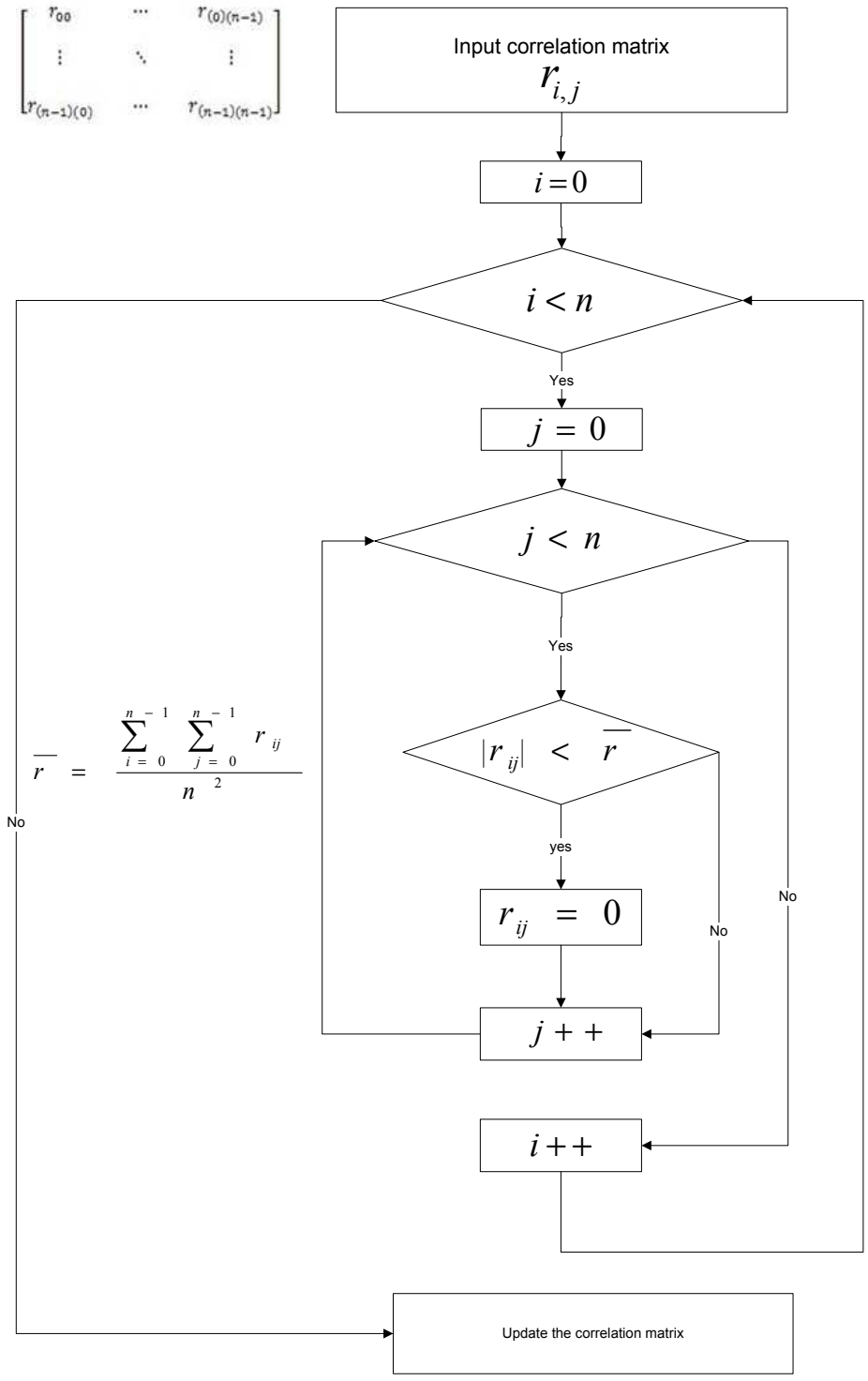


Figure 3.2: Data filtering algorithm

no relation or edge between them in a graph (see Figure 3.2). Based on the above data filtering algorithm, an original correlation matrix is changed into a filtered correlation matrix. Compared to the original matrix without a filtering function, the new matrix removes the uncorrelated data so that the degree distribution tends to a power-law distribution. The changes from the old matrix to the filtered matrix can be displayed in color maps (see Figures 3.3).

3. Drawing a histogram of degree distribution

   In order to verify that the filtered correlation matrix has a power-law degree distribution, we use a histogram to observe the degree distribution in the correlation matrix. This method investigates the degree distribution in the correlation matrix. For example, Figure 3.4 contains two histograms in degree distribution after filtering with a threshold 0.8 and 0.9. The two histograms have the same property that a small proportion of nodes has a high degree, while the vast majority of nodes have a low degree. This property demonstrates that the selective threshold method can be helpful and useful to gain a degree distribution according to a power-law.

3.2 Constructing A Hierarchy

To achieve both faster convergence and better global positioning of nodes in our layout algorithm, we use an undirected graph to construct a hierarchical tree with several levels. The traditional graph visualization tools always overwhelm the user with the increasing complexity of the edges and nodes. The goal of this hierarchy layout algorithm is to break the complex structure and simplify it into a hierarchy structure level by level. Our layout algorithm is developed based on the approach designed by Kumar et al. [6]. A core idea of the two algorithms is to construct a hierarchy tree in which nodes are stratified into separate levels, so that the total degrees in each level are the same. Although stratification algorithms based on the power-law degree distribution have been investigated by Chan et al. [8], that method simply chose a constant number as a graph depth (e.g., namely 3 for all graphs).

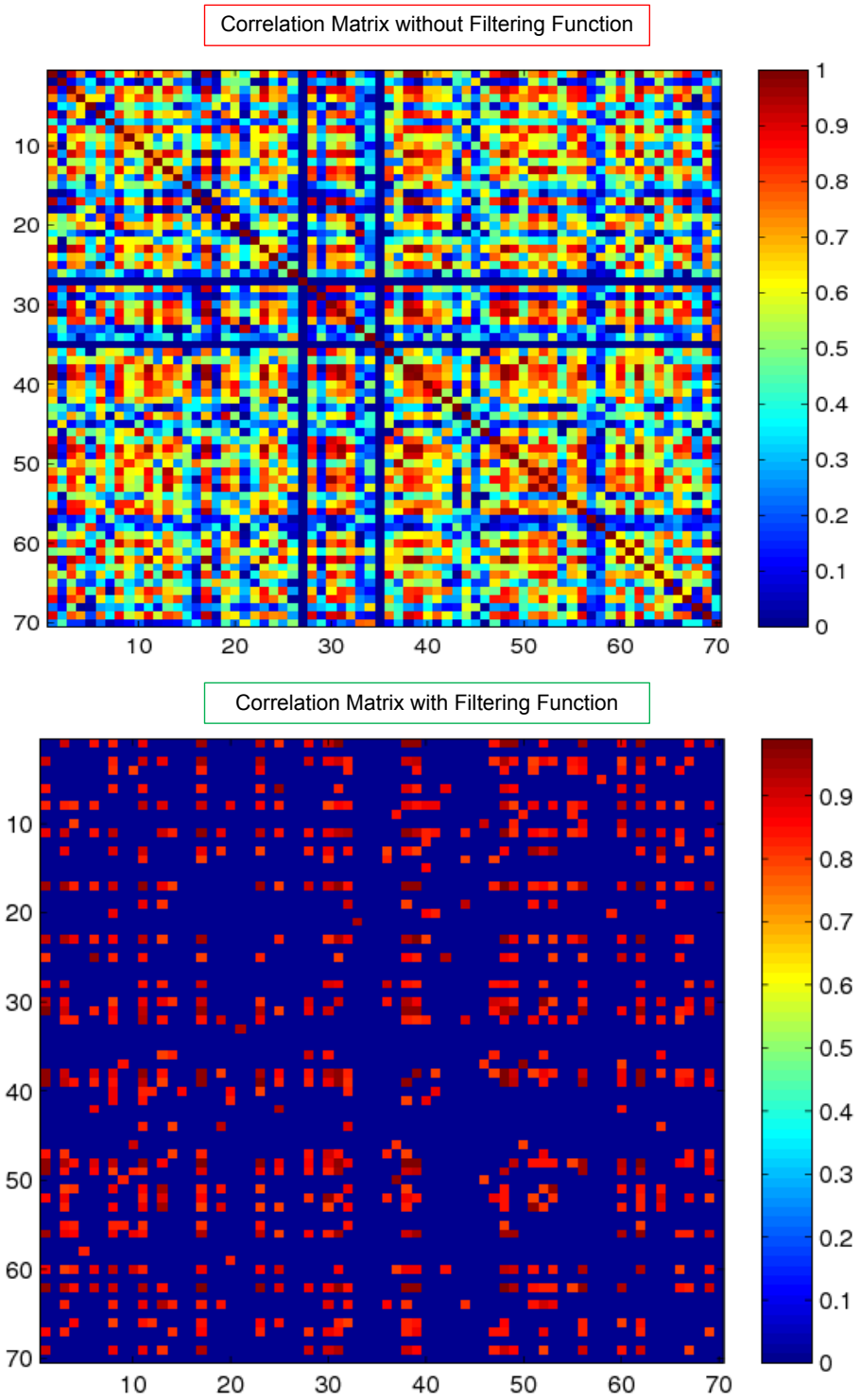


Figure 3.3: Correlation matrix filtering

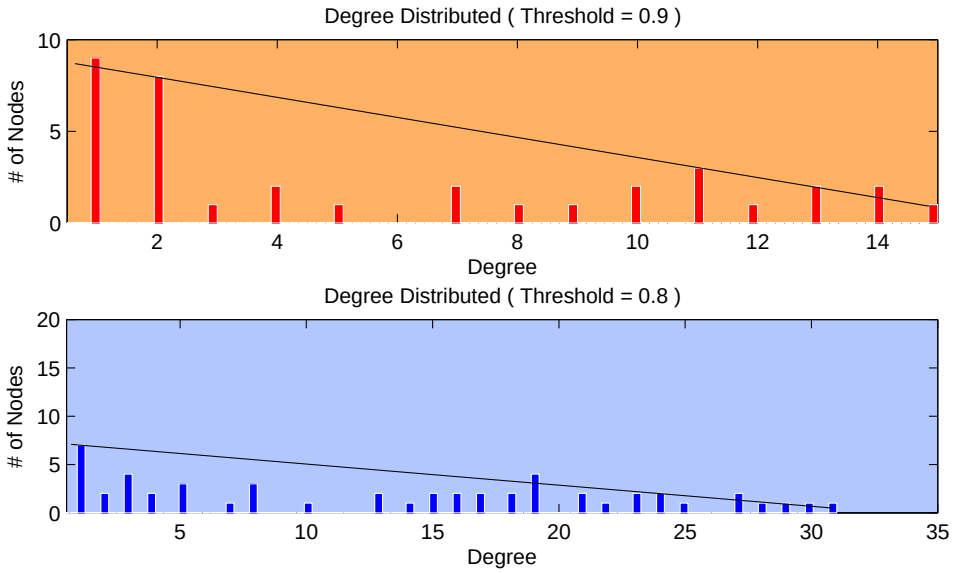


Figure 3.4: Histograms of degree distribution with global average correlations – threshold 0.8 and 0.9

The hierarchy layout algorithm is based on the power-law degree distribution, and we can design a formula to set an adaptive tree depth which depends on the total degree of nodes in the first level. In addition, the hierarchy layout algorithm reduces the complexity of certain formulas and procedures, which improves the performance and speeds up the application.

To construct the hierarchy tree, we undergo three important steps, including sorting the nodes, stratifying the graph, and building family relations. In sorting the nodes, we put the nodes in a queue, and the most important node is placed in the head but the least important node is in the tail. Next, we divide this queue into several blocks which have the same total degree. And meanwhile, we switch a horizontal queue into a vertical tree graph. Finally, a family relationship between any two neighboring levels of the tree is built. Figure 3.5 shows an example for demonstrating the procedures of this algorithm.

3.2.1 Sorting the Nodes by Link Cost

Assume that we are given a graph $G = (V, E)$ with a link cost l_{ij} assigned to each edge (i, j). This thesis displays the data in a graph, where the edge shows the connection between

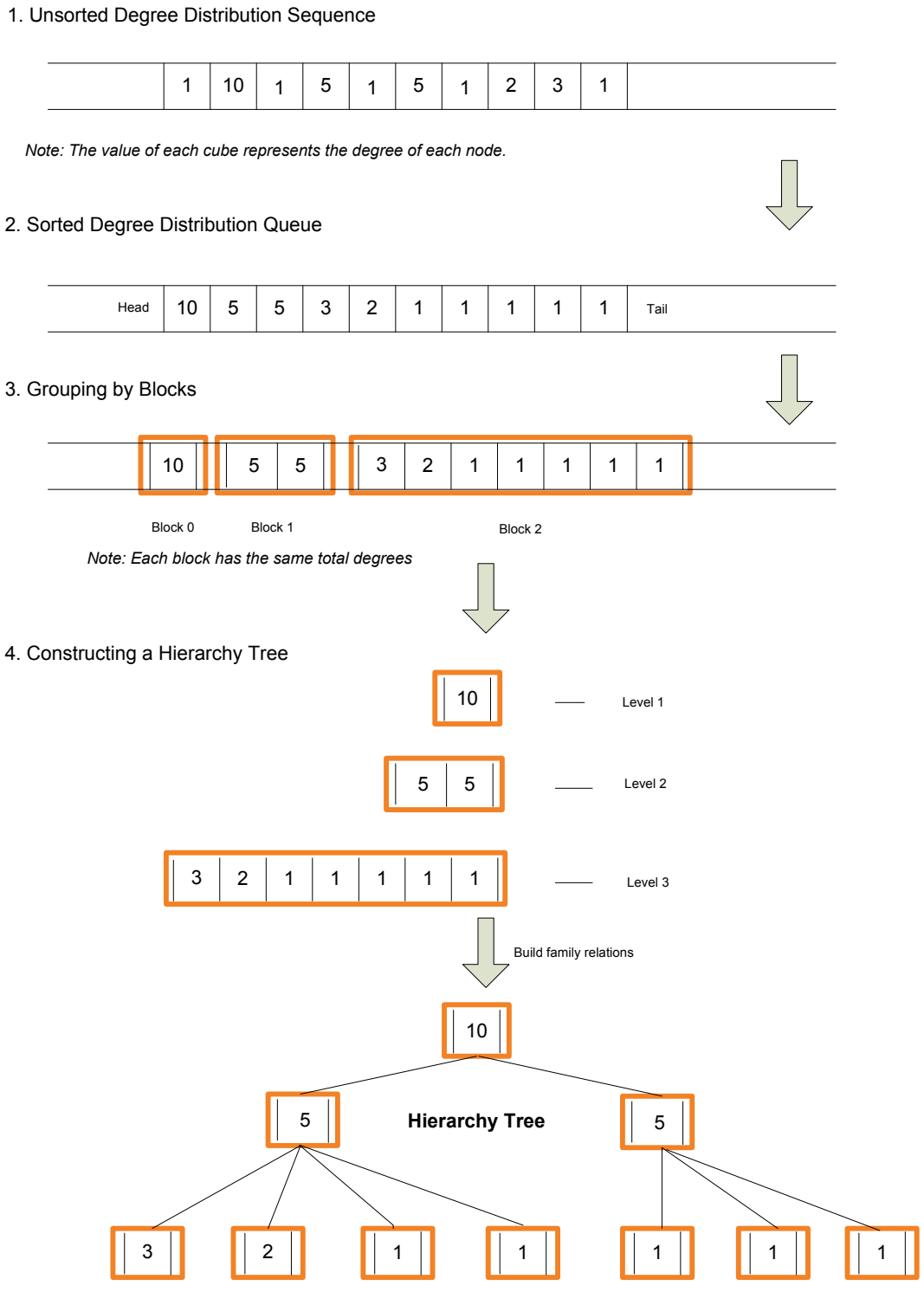


Figure 3.5: The handling procedure in building a hierarchy tree

any two vertices. Here, the total degree of each node is calculated and sorted in descending order. This is saved as a sorted sequence in a queue in which the most important node is a head and the least important node is a tail. In order to create a hierarchy tree, Kumar et al. employed the sorting factor formulation which was inspired by the HITS ranking system [27]. Using the sorting factor, they sorted nodes in an order, that is, the most authoritative and important nodes were placed in the centre. To gain the common target, we design a fast sorting formula to rank nodes. Assuming that each link cost equals the correlation value between two nodes, the sorting factor formulation is represented by:

$$S_i = \sum_{i,j \epsilon V} l_{ij} \tag{3.4}$$

where l_{ij} is the link cost between node i and j. This formula calculates the sum of absolute correlation values for each node. The larger value of sorting factor S_i a node has, the more important and significant positions in the graph structure a node locates. Based on the above formula, we calculate the sorting factor S_i for all nodes in our graph and use a simple and typical sorting algorithm – bubble sort [28] to rank the nodes in descending order. Because of the sorting, we put the significant nodes with larger values in the top position so that the most authoritative nodes are immediately visible to the user.

3.2.2 Stratifying the Graph

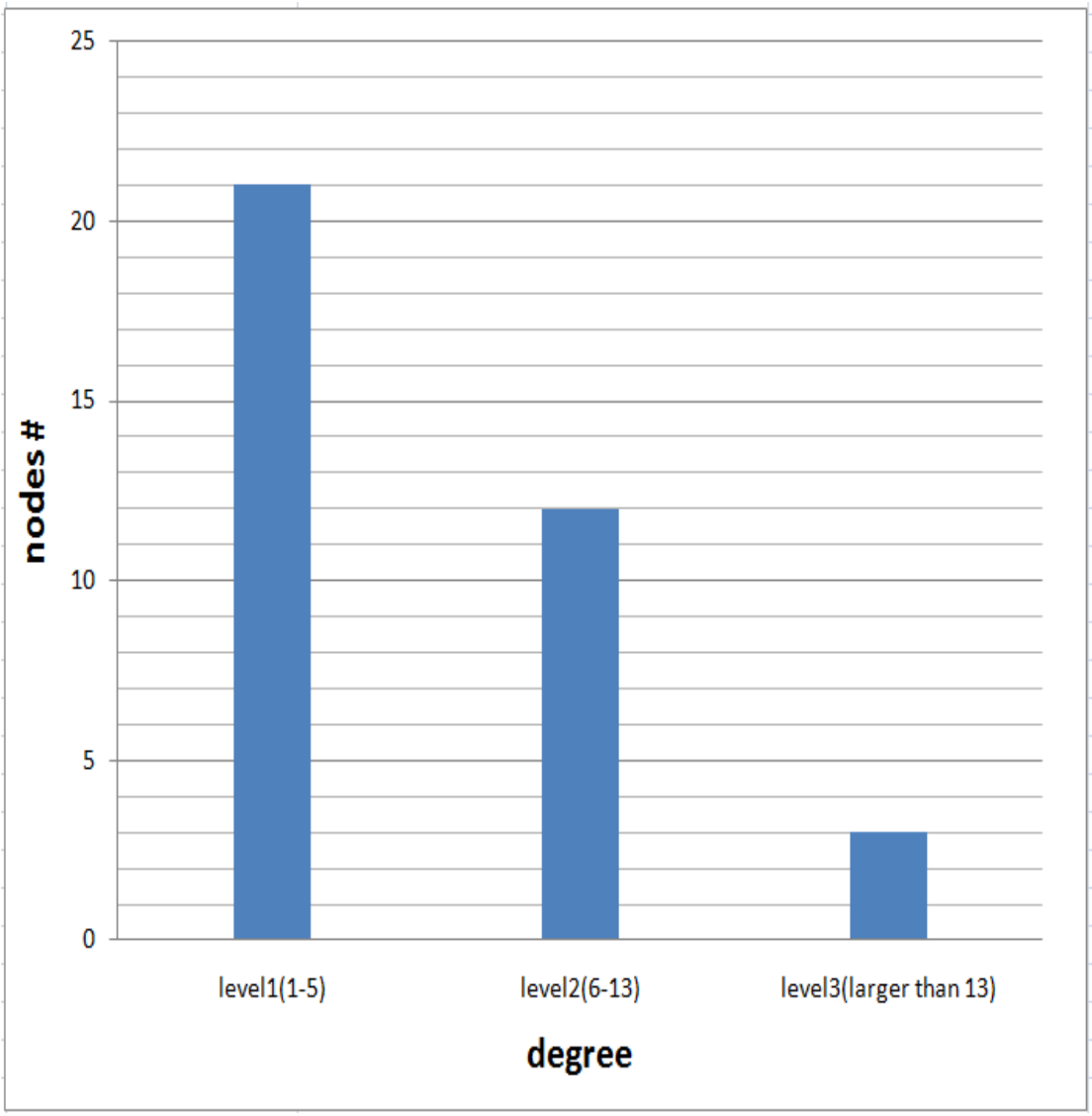
In this part, we will stratify all nodes into different levels in the hierarchy structure. To achieve this goal, this thesis first separates the above sorting queue into several blocks. Each block has the same number of total degrees and the blocks keep the same sequence in the sorting queue. The sorting blocks in the queue are assigned to several levels from top to bottom.

In Chan et al.[8] layout algorithm, three levels are used for all graphs. Kumar et al.[6] develop a method for estimating the depth of the tree depending on the power law exponent. $log_\beta n$ is used to calculate the depth of the tree, where β is an exponent of a power law and n represent the number of nodes in our graph. According to this result, a power law with

exponent $\beta = 2$ provides the best fit for the degree histogram of the S&P 500 graph. However, this exponent of zeta-function (Riemann zeta function [29]) cannot be used in any datasets. The degree distribution is not perfectly matched with the Riemann zeta function; therefore it can not be used in this application. To solve this problem, a new degree distribution is constructed according to a power-law distribution. To construct the new degree distribution, we iterate through nodes in order of authority S_i and place nodes in several groups so that each group has an equal total degrees. According to the degree distribution in the statistic table and graph below, we can directly discover that the degree distribution will be consistent with a power-law. (see Figure 3.6) Depending on the above degree distribution histogram, we stratified all nodes into three levels manually. But, based on the different number of nodes in the first block, we found that the depth of a hierarchy tree was totally different. In other words, the depth of the hierarchy tree relies on the total degrees in the first level. In order to solve this problem, we developed a formula to customize the depth of a tree. Because the total number of degrees in each level would be same in the power-law distribution, we define a formula to calculate the depth of the hierarchy tree. In order to design a flexible and scalable formula, we provide an input parameter in the client side. The client input can decide how many nodes you want to display in the first level. The formula of calculating the tree depth is shown as below:

$$D_{tree} = \lceil \frac{\sum_{i=1}^{n} D_i}{\sum_{j=1}^{k} S_j} \rceil \tag{3.5}$$

where n: the number of all nodes; k: the number of the nodes in the first level; D_i: the number of degrees in the ith node without ranking; S_j: the number of degrees in the jth ranking node in descending order. According to the above formula, we can easily calculate the depth of the tree. Considering the sorting factor, we put the nodes with high degrees into the top level. At this point, our nodes have been positioned in the several levels. In the following step, we will describe the algorithm of building families for all nodes.



Degree Level	Sum of nodes #
level1(1-5)	21
level2(6-13)	12
level3(larger than 13)	3
Grand Total	**36**

Figure 3.6: Degree distribution

3.2.3 Building Families

Due to the above stratification, the nodes have been sorted in different levels. From the above description, we also found that the graph stratification depends on the sorting factors. If a node has a larger value of the sorting factor, this node will be placed in the more significant position which represents its authority and importance. In our hierarchy structure, the first level is the most important and significant place where the clients make a decision. To build family relations in the hierarchy structure, the nodes in the upper level are chosen by nodes in the lower level. A node in the lower level only picks up a sole node in the upper level as its parent. This rule avoids the edge interactions in the hierarchy tree and makes clients clearly distinguish the different subtree or subgroup in the hierarchy structure. Therefore, building the family relations among nodes is a core question in building the relations among nodes in the hierarchy tree. To solve this problem, we develop a parent-finding algorithm.

In order to describe our family tree building algorithm more clearly, several steps and different states are required to illustrate it. According to the power-law distribution, we estimated the depth of the hierarchy structure and we put the nodes into multiple levels with a descending order. For example, we assume that we have four nodes in two levels of our hierarchy tree. At this time, we named the current situation as State 1. The State 1 can be shown in Figure 3.7: In this case, A, B and C have been put in the top level 1, and meanwhile D has been put in level 2. According to our layout algorithm, D will try to find its parent from the top layer. Based on our correlation matrix, D will have three edges with other three nodes normally. Because D only has one parent node according to our algorithm, our layout algorithm will help D to choose a parent node with the highest correlation coefficient value and the highest link cost in the whole structure. This can be defined as State 2 - Case 1 (see Figure 3.8). In State 2 - Case 1, D will choose the best parent C because the value of S_{CD} is the largest one among S_{CD}, S_{BD}, S_{AD}. In addition, we also have a more complicated case 2 in State 2. Let us extend the level of the tree to three. We have two sub-trees in the three levels. One's root is A and the other's is B. The tree structure is shown in Figure 3.9. In the State2 - Case 2, we can observe that the link

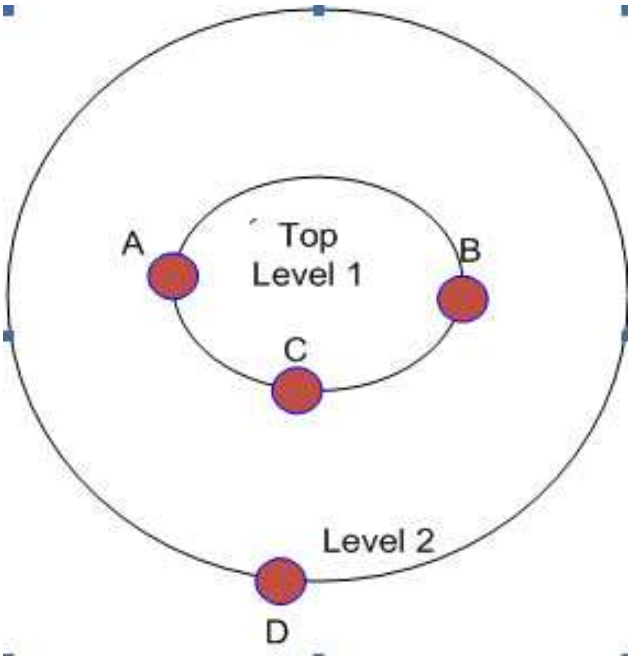


Figure 3.7: State 1 in Building Family Tree

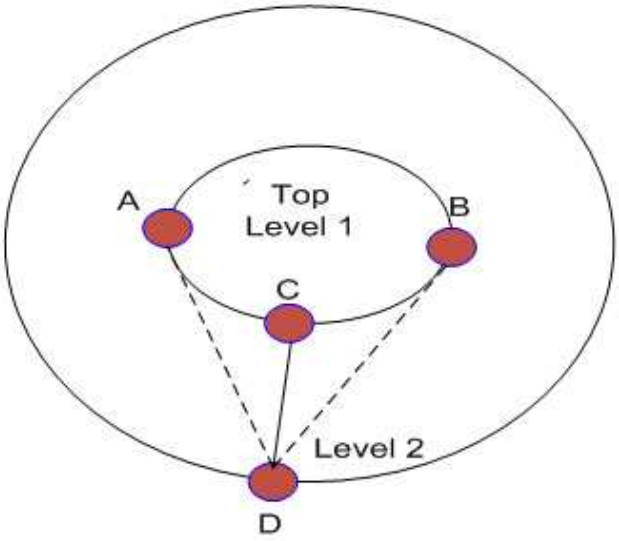


Figure 3.8: State 2 - Case 1 in Building Family Tree

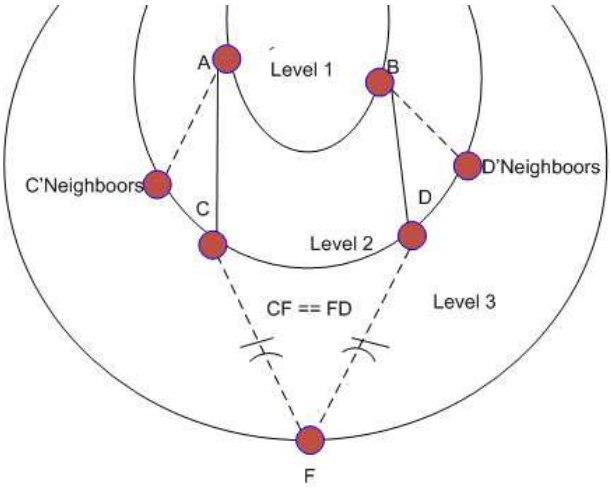


Figure 3.9: State 2 - Case 2 in Building Family Tree

costs of two edges (l_{CF} and l_{DF}) are equal. Although C may have a higher sorting factor or a lower sorting factor than D, we cannot easily assign C with a higher sorting factor as F's parent. In order to choose a precise parent for F, an exhaustive Parent-Finding algorithm is developed. Only when the link cost in two edges is the same, will this algorithm be valid. We choose a parent relying on both the direct link cost between two nodes and the undirect other link costs. The pseudo code of this Algorithm is shown in Figure 3.10.

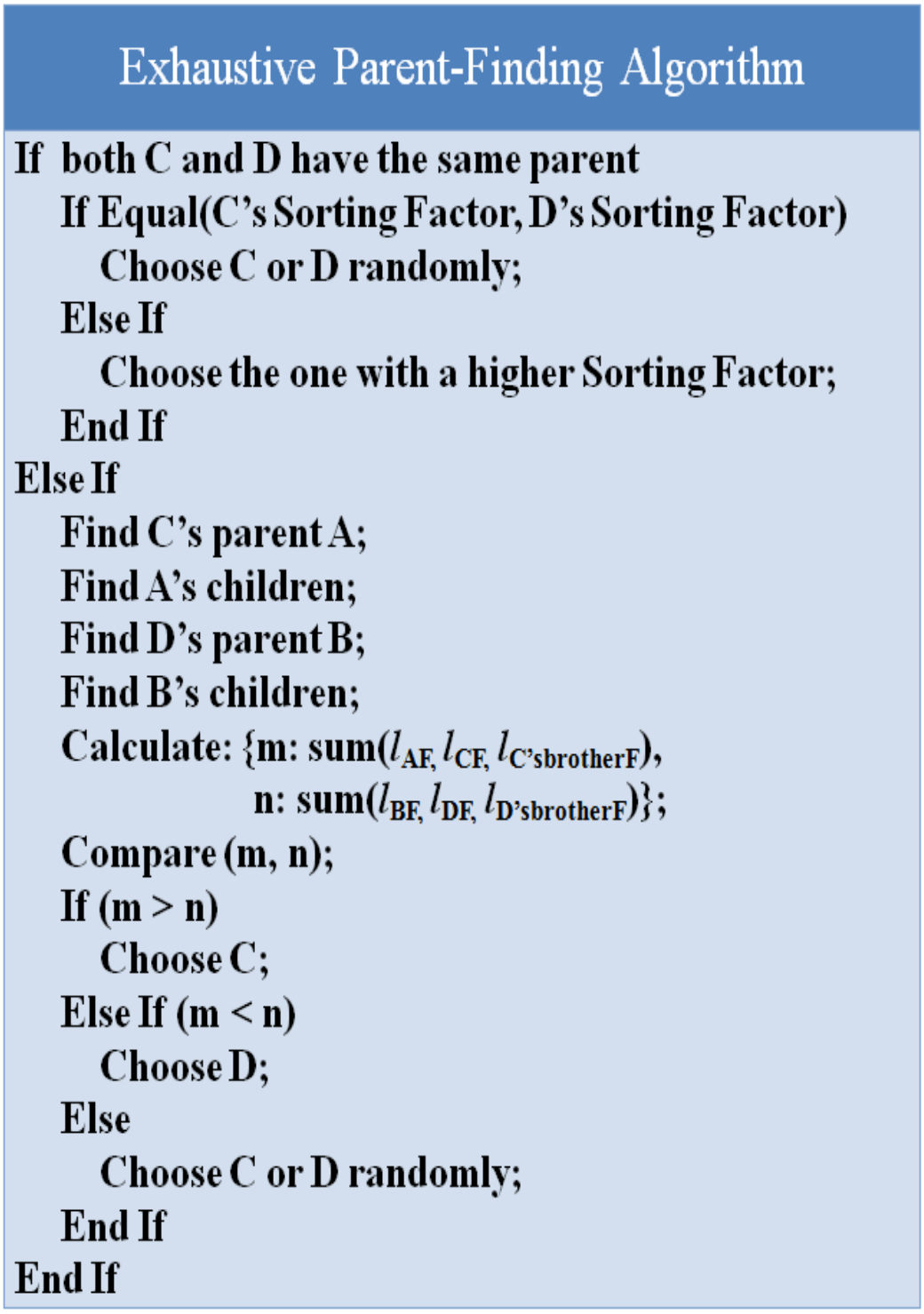


Figure 3.10: Exhaustive Parent-Finding algorithm

Chapter 4

Multi-Functional Graph Visualization System

The main goal of our visualization system is to implement our hierarchy algorithm, which visualizes the financial time-series data and help users to understand the data in a hierarchy algorithm. In addition, a good graph visualization system can achieve both faster convergence and better global positioning of nodes, and meanwhile the layout result cannot overwhelm users due to complexity. Based on the above standard, our graph visualization system was developed using the BirdEye Information Visualization and Visual Analytics Library [30], and incorporated interactive graph visualization and navigation techniques into our system application. In this chapter, we mainly focus on introducing the structure, functions and features in our graph visualization system, particularly emphasizing the specific and unique features in our system.

4.1 System Structure

The three-tier development architecture is one of the most popular architectures in the software development. The three-tier architecture is widely utilized in the development of most online web application systems. Figure 4.1 will outline the content and architecture of a typical three-tier system. A three-tier system mainly consists of three tiers, including data tier, logic tier and presentation tier. Apart from the usual advantages of modular software with well-defined interfaces, a three-tiers architecture brings a lot of benefits to

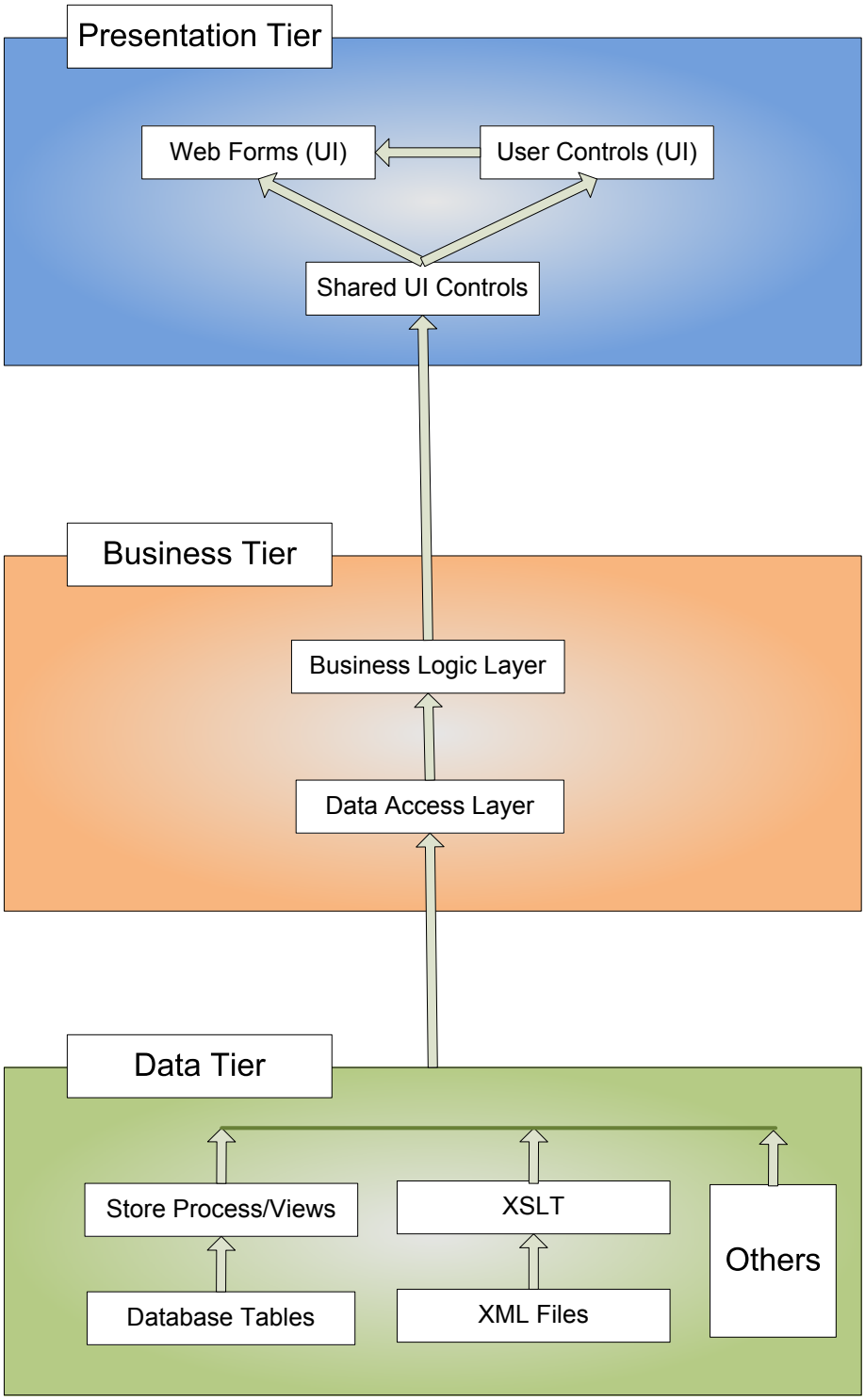


Figure 4.1: Typical outline of 3-tier architecture

the development and maintenance of a software system. For example, the change of a software system in the presentation tier only affects the user interface code. We utilizes the 3-tiers architecture to design and develop our visualization system. Figure 4.2 shows our core development architecture implemented by the three-tier architecture. Our financial data visualization tool was developed by using Java programming language and flex action scripts. Java application in the data tier provides data access functions to help querying the data from the database. In addition, Java programming is utilized in the logic tier to implement our novel hierarchy layout algorithm. In Figure 4.2, we can clearly recognize that Java programming focuses on completing the tasks in the data tier and logic tier. On the other hand, we choose Abobe Flex Builder as our development platform, in which our visualization results are displayed. In the current web application fields, Adobe Flex is a new medium specifically created for designing and developing Rich Internet Applications(RIAs). RIAs are a new breed of applications that break out of the constraints of traditional web and desktop environments to provide a more fluid, information-focused user experience. That is why we use it on the demo.

4.2 Multi-Functional Visualization System

Each graph visualization system will be customized with some new features because of specific applications. Our multi-functional graph visualization system can be used in many general cases, where the data format is consistent with the designed format. The number of nodes in the visualization system will be limited to one thousand. The visualization system mainly concentrates on designing a few unique functions and features for the small or medium datasets. Our focus is not on improving the speed or some problems raising from the large number of nodes. With a thousand nodes, our system would not have a long delay in displaying the graph results like the other visualization systems.

The multi-functional visualization system includes functions and features, such as integration of many graph layout algorithms, and animate interaction and navigation. In the coming section, we will discuss details about those new features.

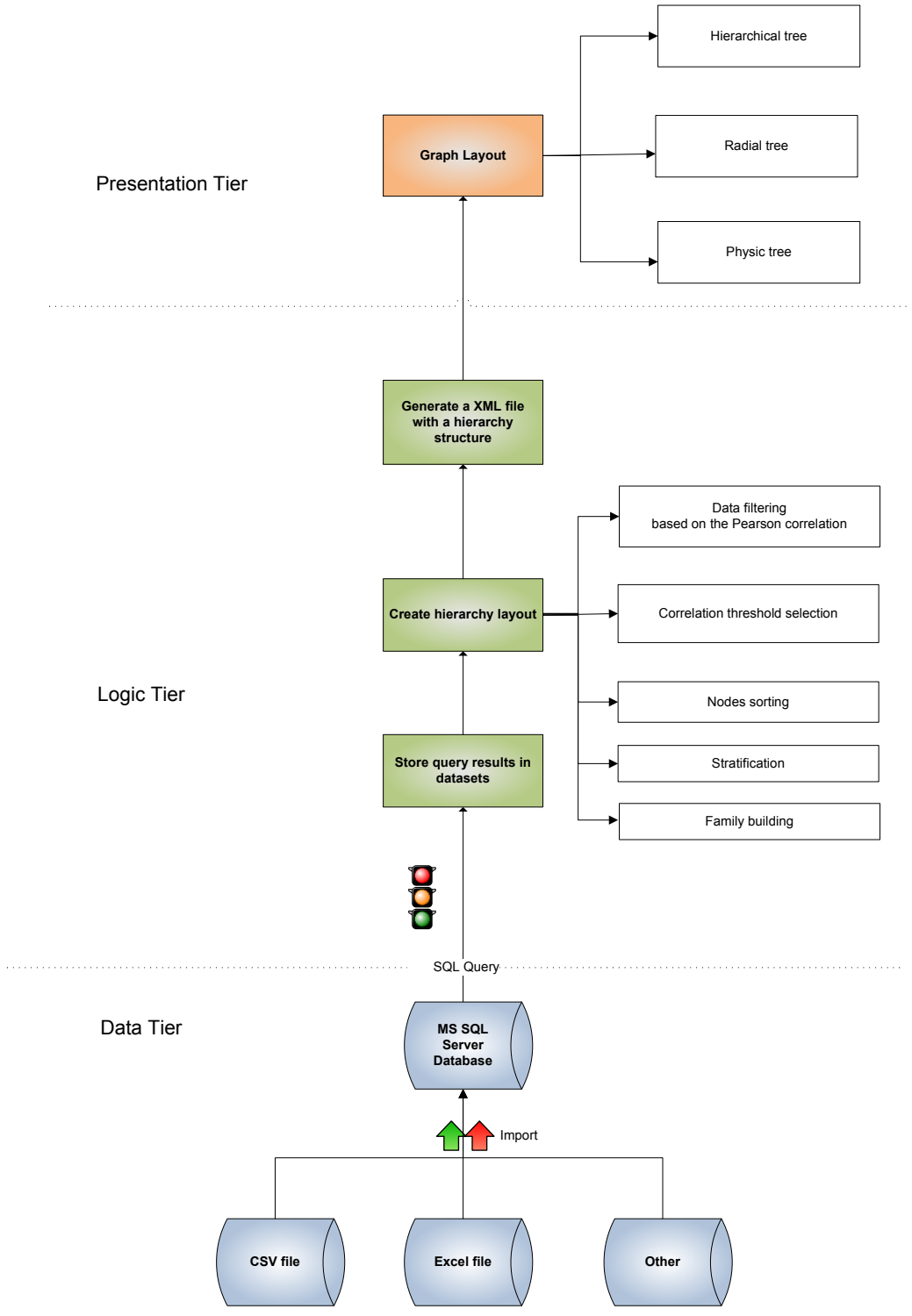


Figure 4.2: New application of 3-tier architecture

4.2.1 Implementation of Many Graph Layout Algorithm

We implement and integrate many kinds of graph layout algorithms, including radial view and hierarchical layout algorithm etc. The main goal is to make users understand the abstract structure, to discern the relations in the data, and to notice the changes in different layout views. Different views will generate different interactions with users, while diverse user communities have their own traditions, habits, or requirements, and an application system cannot impose one single view on its users. Therefore, to achieve this target, a lot of graph layout algorithms have been implemented in our system.

Although our system contains different kinds of graph layout algorithms, the radial view algorithm is the focus of layout algorithms in this thesis. The graph visualized in the radial view can demonstrate our hierarchy layout method directly. In the radial view layout algorithm, we treat the graph as a tree rooted at the focus node. We determine the parent-child relationships in the tree by performing a traversal of the graph, starting from the focus. In a radial layout, the focus is placed at the center of the graph and lay out the other nodes around this focus. A straightforward method for laying out the other nodes, called "radial drawing" in [9], is used here. Nodes are arranged on concentric rings around the focus node. Each node lies on the ring corresponding to its ranking order. The angular position of a node on its ring is determined by the sector of the ring allocated to it. Each node is allocated a sector within the sector assigned by its parent, with size proportional to the angular width of that node's subtree. This is described as "radial placement" in [31], where all the nodes are the same size, and so the angular width of a node's subtree is the number of leaf nodes among its descendants. Based on the hierarchy layout created in chapter 3, we place all nodes into different layers in order to demonstrate the relations among the nodes. According to the nodes distributions in the hierarchy layout, we can easily calculate the angular width of its child subtrees, and allocated all nodes in the subtrees to this sector. Figure 4.3 can illustrate the principle of sector allocation, which assigns the angular to each sub-sector according to the number of nodes in the subtree.

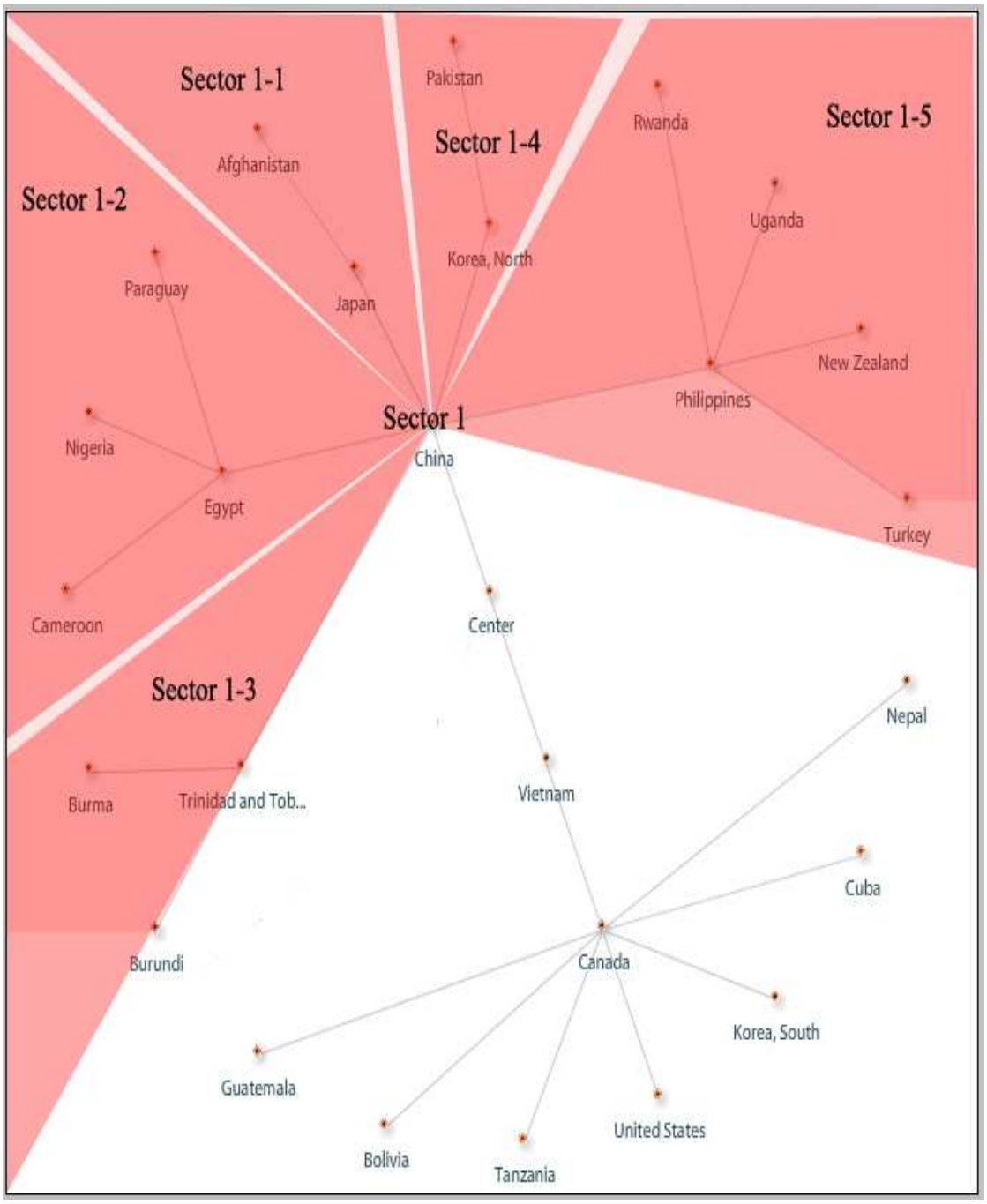


Figure 4.3: Sector allocation in a radial view graph

4.2.2 Interaction and Navigation

A good system have to offer a wide range of tools in order to make the exploration of a graph easy, or even possible. In some cases, the user has to move around in the information space for exploring details or hiding unnecessary parts of a graph. In this situation, interaction and navigation facilities are essential in information visualization.

Animated Zooming and Scaling

Zooming and scaling are traditional navigation skills in visualization technologies. They are quite indispensable when a large graph structure is explored. Zooming can take on two forms: Geometric zooming and Semantic zooming [1]. Geometric zooming simply change the size of graph content. Semantic zooming means that the information content changes and more details are shown when approaching a particular area of the graph. In most cases, Geometric zooming is widely used in traditional visualization systems, including resizing the graph layout (such as minimization and maximization).

In this thesis, we developed a novel zooming and scaling method which is cooperated with animating separation of the degree. In our visualization system, zooming and scaling are not only focused on resizing the graph nodes or minimizing and maximizing the graph layout, but they also can perform the zooming and scaling by reducing and increasing the degree in a tree graph. For example, we have a concentric radial tree with four degrees. When a user wants to zoom into smaller degrees or zoom out into larger degrees, our graph visualization system will automatically change the view with different separations of the degree and animated this transition progress (see Figure 4.4).

In some cases, a user selects a visible node to become the new focus in order to explore the graph. The new layout tree is found by performing a breadth-first search from the new focus, thereby computing the distance from the new focus to each node. The new layout is determined by assigning each node to the appropriate ring and allocating angular sectors of rings as discussed above. At this time, we use the animation technology to track the transits action from one view to another view and also try to keep the same position for selected

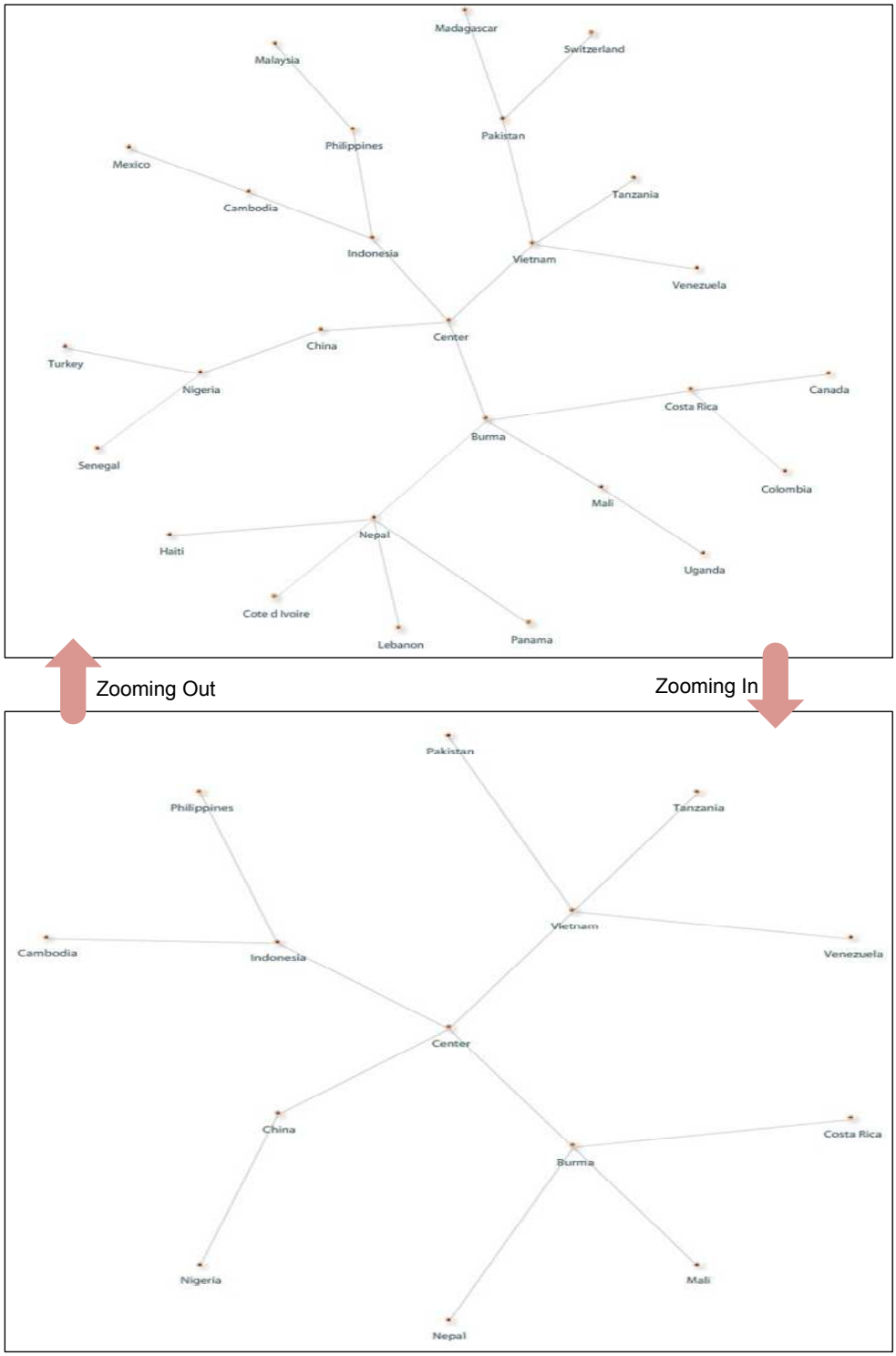


Figure 4.4: Zooming in with the separation of degree from degree 3 to 2

nodes' children (see Figure 4.5). This new animation technique was described by Yee et al. [32], for supporting interactive exploration of a graph. The core contribution of this method is to animate and keep the transition to a new layout when a new focus node is selected.

4.3 Graphic User Interface Design and Implementation

Graphic User Interface(GUI) Design and Implementation is a very important part in an interactive visualization system. A good visualization system will be easy to utilize and generate an aesthetically pleasing result for users. Additionally, an ideal visualization system will provide some customized configurations to users and let them freely change the graph with different conditions. We created the GUI components depending on the above discussion of an ideal visualization system. But a rational GUI design is the first step in the development of a visualization system. The next section described details about the GUI design and implementation of the interactive visualization system.

4.3.1 Graphic User Interface Design

The GUI layout of our system can be divided into three parts: the data source selector, the graph visualization controller, and the graph visualization display(see Figure 4.6). The data source selector can assist clients to navigate and select their interesting data. The graph visualization controller concentrates on controlling the graph visualization display, and providing the layout control, the nodes style control, the edge style control, the edge label style control, and the view control (see Figure 4.7). The layout and view control are the core parts of the design in the graph visualization controller. Many kinds of graph layout algorithms are developed and implemented in the layout control component, which contains radial view (concentric radial tree, parent center radial tree, single cycle radial tree, and hyperbolic tree), hierarchical tree layout (top-down, bottom-up, left-right, right-left), physics layout (force-directed), etc. In the view control, our system implements the new feature, that is, animating zoom in and out with degrees of separation and smoothly switching between

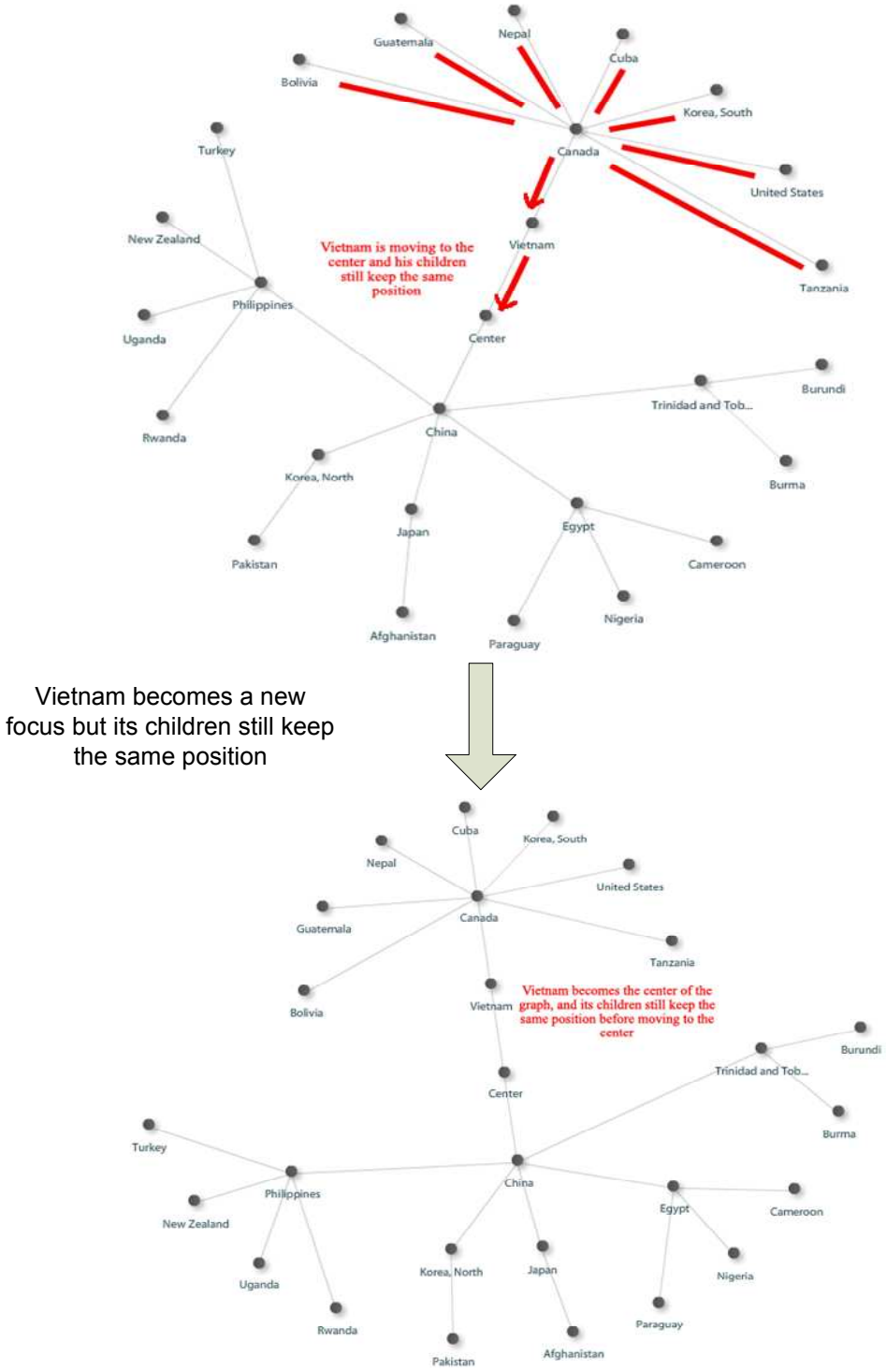


Figure 4.5: Animated transition and consistent position

different layout algorithms. This graph visualization system is implemented with the most popular graph layout algorithms and give users an impressively pleasing graph result. In other words, our data visualization tool can help users visualize the data in various types of graph figures. Although the different graph layouts will generate different effects and results, the user can always find an appropriate graph layout to describe the relationships in the data. All of the graph layout algorithms assist us to build a graph in a variety of perspectives, and meanwhile these algorithms help to contrast which layout is better in the different applications. In the following chapter, we will concentrate on analyzing and comparing the result in different layout algorithms.

4.3.2 Implementation

Our graph layout system is developed based on *BirdEye Library*, which is a community project to advance the design and development of a comprehensive open source information visualization and visual analytic library for Adobe Flex. The actionscript-based library enables users to create various data visualization interfaces for the analysis and presentation of information. In order to adapt to our new application, we create a new visualization tool based on the *BirdEye Library*. The big difference from many current graph visualization tools is that our system is an online interactive system. Implementation of this online interactive visualization system can be divided into client side and server side. In the server side, the server Java application is developed in the J2EE platform in which the novel hierarchy algorithm is implemented and the hierarchy results are stored in the XML file. In the client side, the flash web application is developed in the Flex Builder platform in which the hierarchy structure is displayed in the different views. In the GUI section,the new features of this visualization system have been introduced and described.

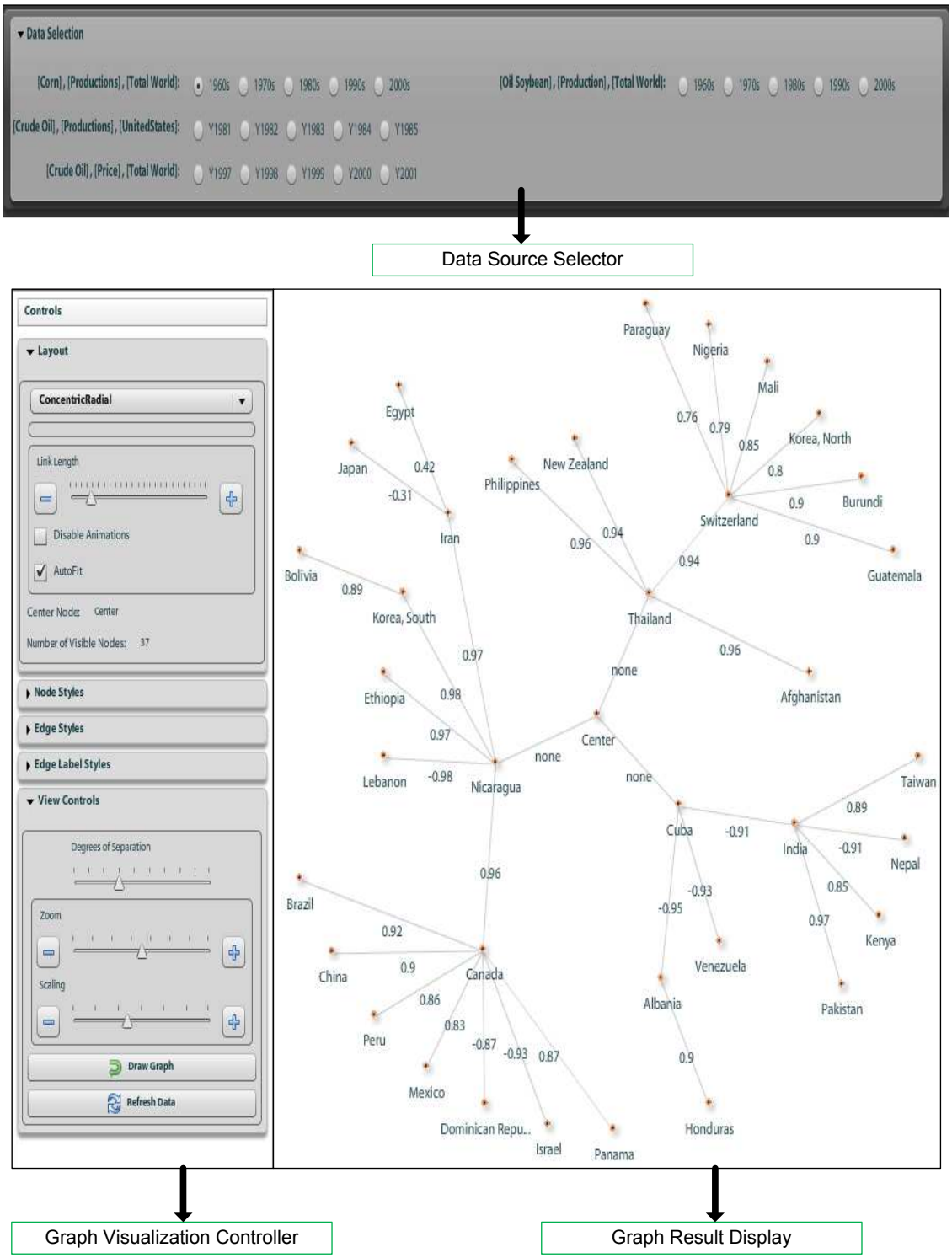


Figure 4.6: GUI Layout of graph visualization system

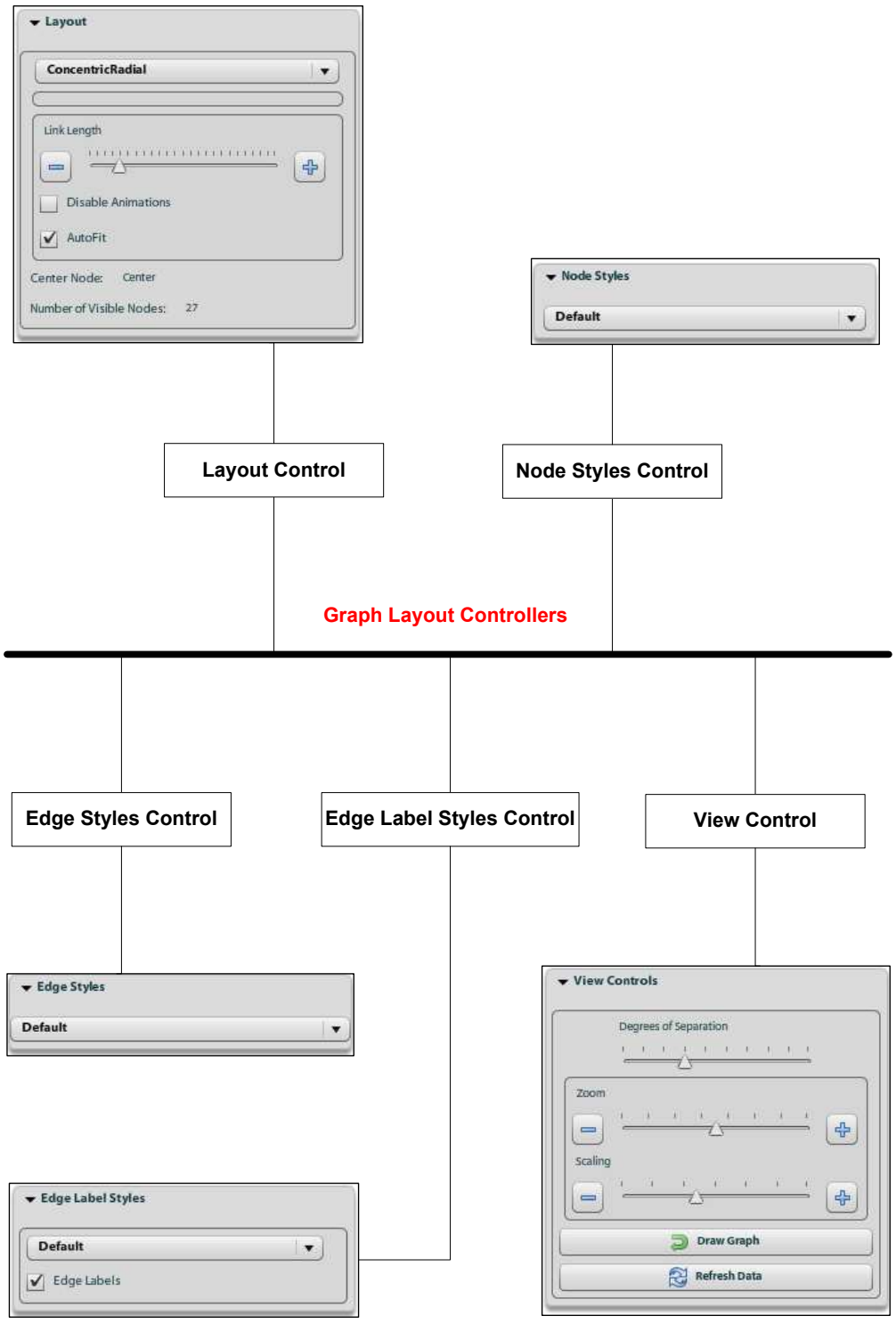


Figure 4.7: Structures of graph visualization controller

Chapter 5

Applications

The main feature of the visualization system is to quickly extract the relations in the financial time-series data, and help people to see the changes of a graph layout in the different time periods. In order to demonstrate benefits of our visualization system, financial time-series data were analyzed. For example, when using our visualization system to display world crude oil price data, the graph result can testify that the price fluctuation of crude oil with OPEC or non-OPEC countries is highly correlated, but that is not strongly correlated between OPEC and non-OPEC countries. In the coming sections, we will present many new applications in our visualization system. Before discussing the details about different applications, we have to clearly understand our data and know how to organize the data.

5.1 Data Organization

There are two main goals in the data organization. One is to eliminate the redundance in the raw data, and the other is to handle the missing data. Eliminating the data redundance can reduce the complexity of the data structure and improve the performance of data retrieval. The benefits of handling the missing data can preserve the data integrity.

Our financial time-series data sources are from different public websites, such as the weekly data of world crude oil price which is from the United States Energy Information Administration(EIA) website [33] and the yearly data of fundamental agriculture commodity which is from the United States Department of Agriculture Foreign Agricultural Ser-

vice(USDA) website [34]. Because each data source has an individual format, we have to re-organize the data according to an uniform standard and demands. To clearly demonstrate the method of data organization, we take USDA yearly data as an example to introduce the data organization in the coming discussion.

The raw data is downloaded in an Excel format. The raw data were imported into Microsoft SQL Server which is a relational database server produced by Microsoft. In the MS SQL Server database, the USDA data is stored in a relational data table, which contains five properties: commodity, attribute, year, country, value. For instance, one record in USDA data table can be described as: commodity – corn; attribute – production; year – 1999; country – China; value – 10,000. Compared to S&P 500 stock data which is composites of two main properties – price & volume, the USDA data has a relatively complex data structure, which leads to a long response time of the data retrieval. This problem seriously affected the performance of the web application system because of a low speed in the data retrieval. The solution of solving the problem is reducing the complexity of the data, which means that we partition a large and complex data tables into several small and simple data tables. This method is called data table partition, help reducing the redundance in the data table. According to the data table partition, we improve the speed in the data retrieval and enhance the performance of our visualization system.

In addition, the raw data from public websites exists some missing points in a continuous data sequence. Comparing the similarity between two data sequences, the Pearson's correlation method requires the two consequences with the same length. The missing data points make the length of two data sequences different, which causes the Pearson's correlation method invalid. In order to solve this problem, we propose an method to estimate the missing data and fill an adaptive value in the blank position.

Based on the above discussion, data organization can be divided into two parts:

- Eliminating the redundance in our database.

  First of all, we have to remove the repeated data record from the original table. And then, we also have to re-design the data structure to improve the response time of data

retrieving.

- Preserving data integrity.

  Because the raw data series often have the missing data in some specific time points. In this point of view, we develop a method to estimate a value of the missing data.

5.1.1 Data Table Partition

Data table partition is dividing a data table into several sub-tables, which simplifies the data structure and improve the performance of data query. In other words, data table partition does not only reduces the redundance in our database but also optimize the speed of data retrieval. The implementation of data table partition is different according to different applications. But, the processing steps of table partition is the same. For example, in order to investigate USDA corn productions in all countries of the world, we have to prune our data table based on the commodity field and attribute field. The raw USDA data table is first divided into several commodity data tables, such as table_corn, table_coffee, etc. The commodity data table is then separated into several attribute data table, such as table_corn_production, table_corn_import, table_corn_export, etc (see Figure 5.1). Through trimming data table, we generate a more specific data table with a small size. And meanwhile, we also reduce the dimensions of data table which has less columns than the raw data table, so that the complexity in the data structure is reduced.

5.1.2 Missing Data Handler Method

The goal of handling the missing data is to preserve data integrity, which makes sure that the Pearson's correlation formula can be used to calculate the correlation coefficient between two data series in the same time periods. We take the yearly data of world corn production in the past ten years(2000-2009) as an example. In general, each country has ten data points to represent the past ten years' corn production. In the situation of the missing data, some countries lose one or two data points in the past ten years. The data sequences with the missing data cannot be calculated as the correlation coefficient according to the rules in

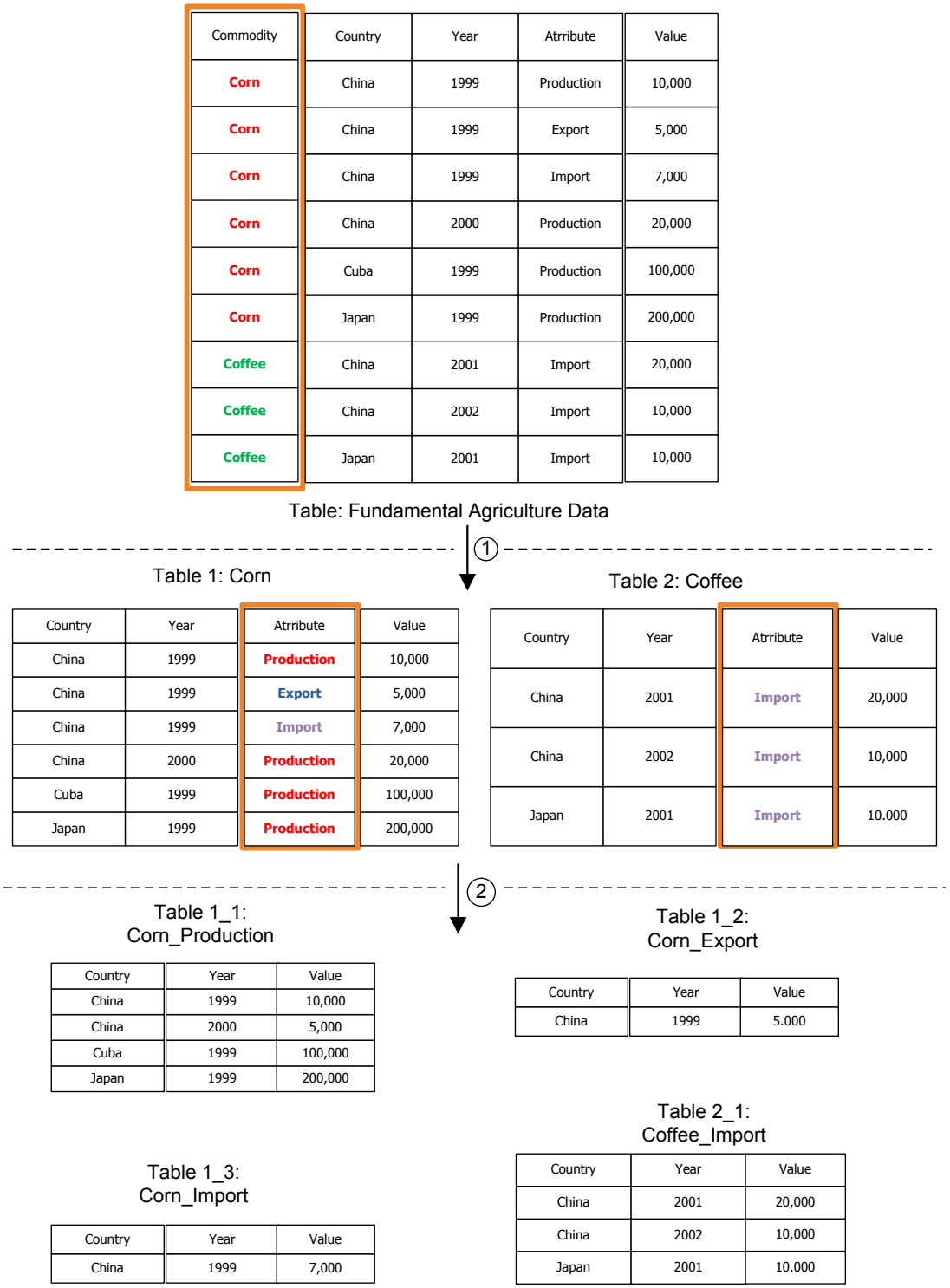


Figure 5.1: The Process steps of table partition

Pearson's correlation. To solve this problem, we develop a method to handle the problem in missing data. If the number of missing points is larger than two within ten points, the entire data series will be ignored. Figure 5.2 shows the flowchart of the method in handling the missing data. In this method, the position of missing data will be discovered firstly. If the missing data has two neighbours (front and back) in the data sequence, the average value between neighbours will be considered as the value of missing data; If the missing data only has one neighbour, the value of missing data is the same as it's neighbour's; Otherwise the entire data sequence will be ignored.

5.2 Unique System Features Applied in Fundamental Agriculture Data

Based on the organized data, we apply our visualization system to display the fundamental agriculture data in the hierarchical graph. Depending on this application, we can obviously find that the new features of the visualization system make user see the graph in the diverse perspectives. To start with, the layout switch with degree separations is one of the most signification features in our system. Figures (5.3, 5.4, 5.5) show the distribution of world corn productions from 1960 to 1969, and the three radial trees have the different degrees (the degree of a graph = 1, 2, 3). In Figure 5.3, the top nodes of the radial tree are highlighted, and the center nodes are also the root nodes of the subtrees. Switching to Figure 5.4, we can see that the nodes in the second level find their parent nodes from the upper level. Following the same rule, Figure 5.5 also shows that the nodes in the third level locate the position and find their parents nodes from the second level. The transitions of degree separations in different graph layouts can help clients understanding how to construct the hierarchy algorithm.

Analyzing Figure 5.5, we can find that the node(Cuba) is placed in the first level and it can be considered as the most significant and important country in the corn production from year 1960 to year 1969 (1960s). Because there are a high proportion of countries correlated to Cuba, which connects to several subtrees with a large number of nodes. Based on the

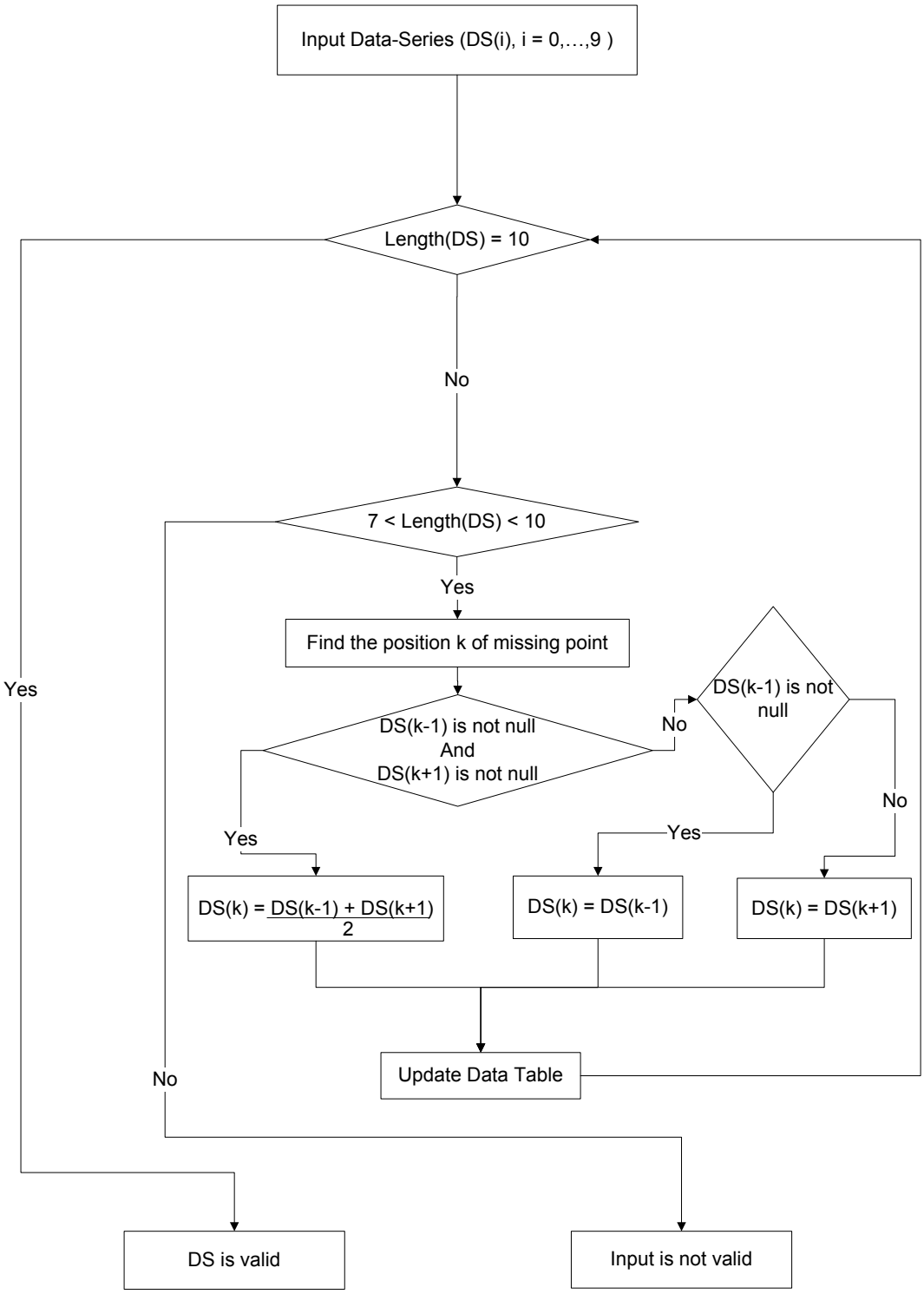


Figure 5.2: Missing data handler

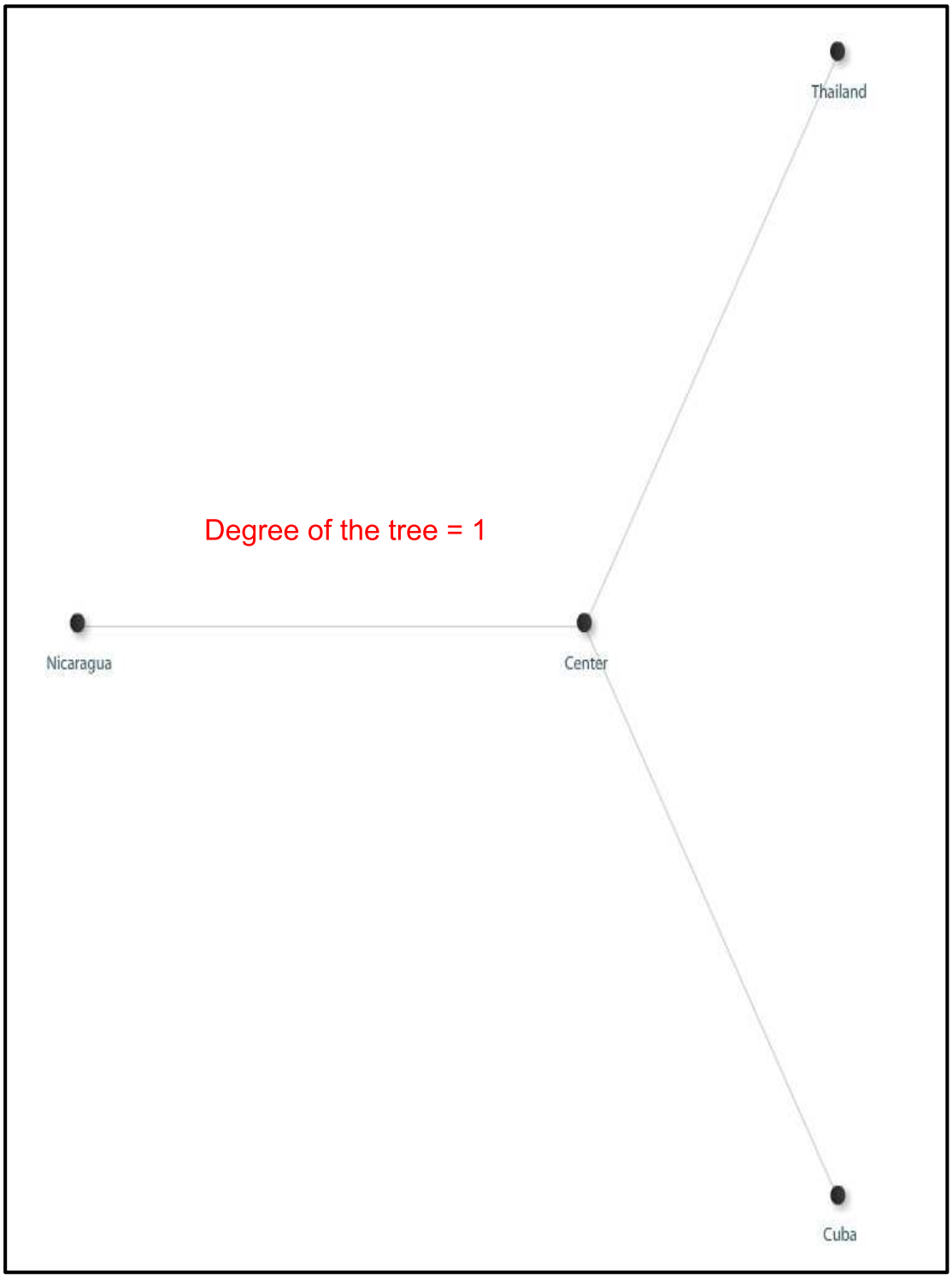


Figure 5.3: The distribution of world corn productions from year 1960 to year 1969. (the degree of a graph = 1)

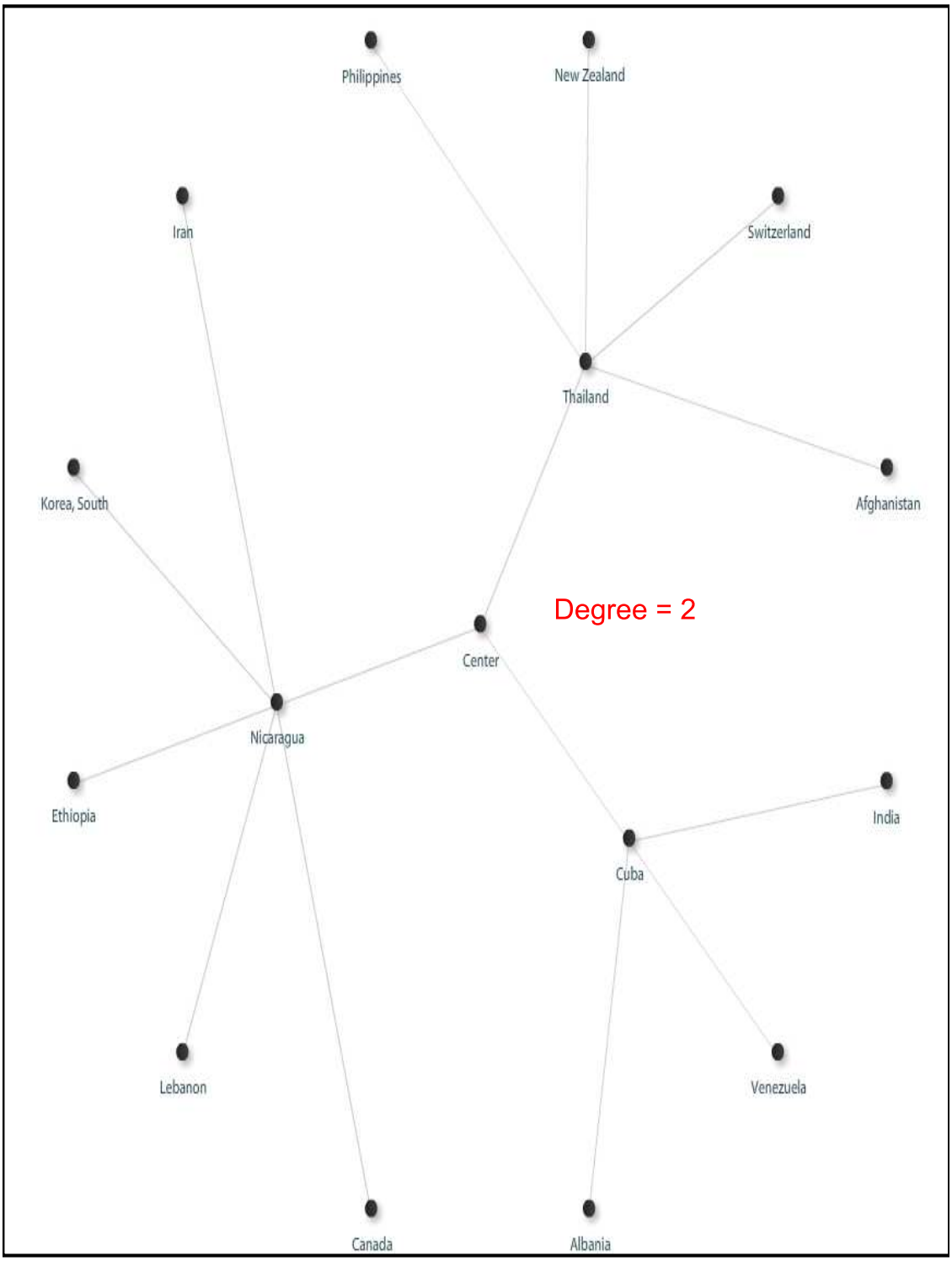


Figure 5.4: The distribution of world corn productions from year 1960 to year 1969. (the degree of a graph = 2)

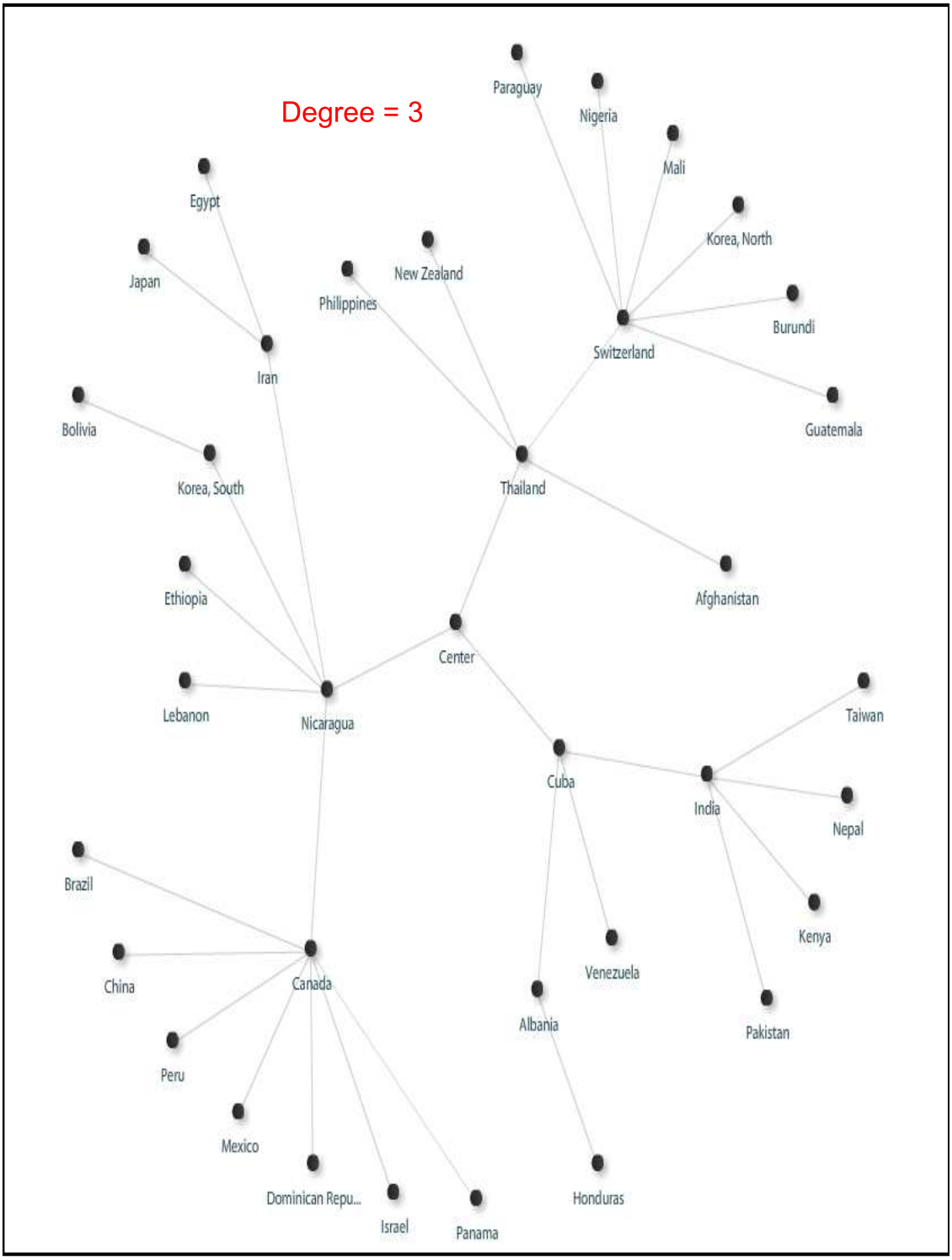


Figure 5.5: The distribution of world corn productions from year 1960 to year 1969. (the degree of a graph = 3)

analysis of an entire radial tree, we can discover the most significant nodes. Besides, the change in the position of center nodes leads us to find that the country distribution changes during the different time periods. Figure 5.6 displays the changes in the country distribution of the corn production from 1960s to 1970s. In Figure 5.6, we can obviously discover that the position of the center node Cuba in the 1960s tree is switched to Canada in the 1970s tree. In 1970s, Canada instead of Cuba is strongly correlated with other countries, and the radial tree puts it into an obvious position.

Additionally, the graph layout system has a great feature to display the relations between each two nodes. This feature makes users more clearly verify if the relationship is positive or negative. Regarding to the graph, it is best way to add an edge label on the link between each two nodes. To implement this feature, we put a symbol on the edge, which represents the negative or positive relationship between each two nodes. Figure 5.7 displays the symbol on the connection according to the correlation relationships. In the Figure 5.7, you can find out three different label symbols or words attaching the edge. The words *none* means that there is no real meaning between them; the symbol "+" means that there is a positive relation and the symbol "-" means that there is a negative relation. The edge label "+" indicates that there is the same trend between two nodes in a link, and the edge label "-" denotes that two nodes in a link have an opposite trend in the same time periods. To testify whether the symbols display a correct relation in the link, we will randomly pick up two countries with a positive or negative connection. Figure 5.8 consists of two charts: negative correlation and positive correlation. The negative correlation chart displays an negative linear correlation in the corn production between Cuba and India from 1960 to 1969, and the positive correlation chart also shows a positive linear correlation in the corn production between Canada and Nicaragua from 1960 to 1969. Compared to Figure 5.7 which is the hierarchy layout result in the corn production during 1960s, we can observe that the edge labels is consistent with the linear correlation between the nodes. Figure 5.7 shows the layout graph with edge labels which represent the negative or positive linear correlation in a pair of nodes. Alternately, instead of a symbol ("+" or "-"), the real correlation value attaching to the edge is a more

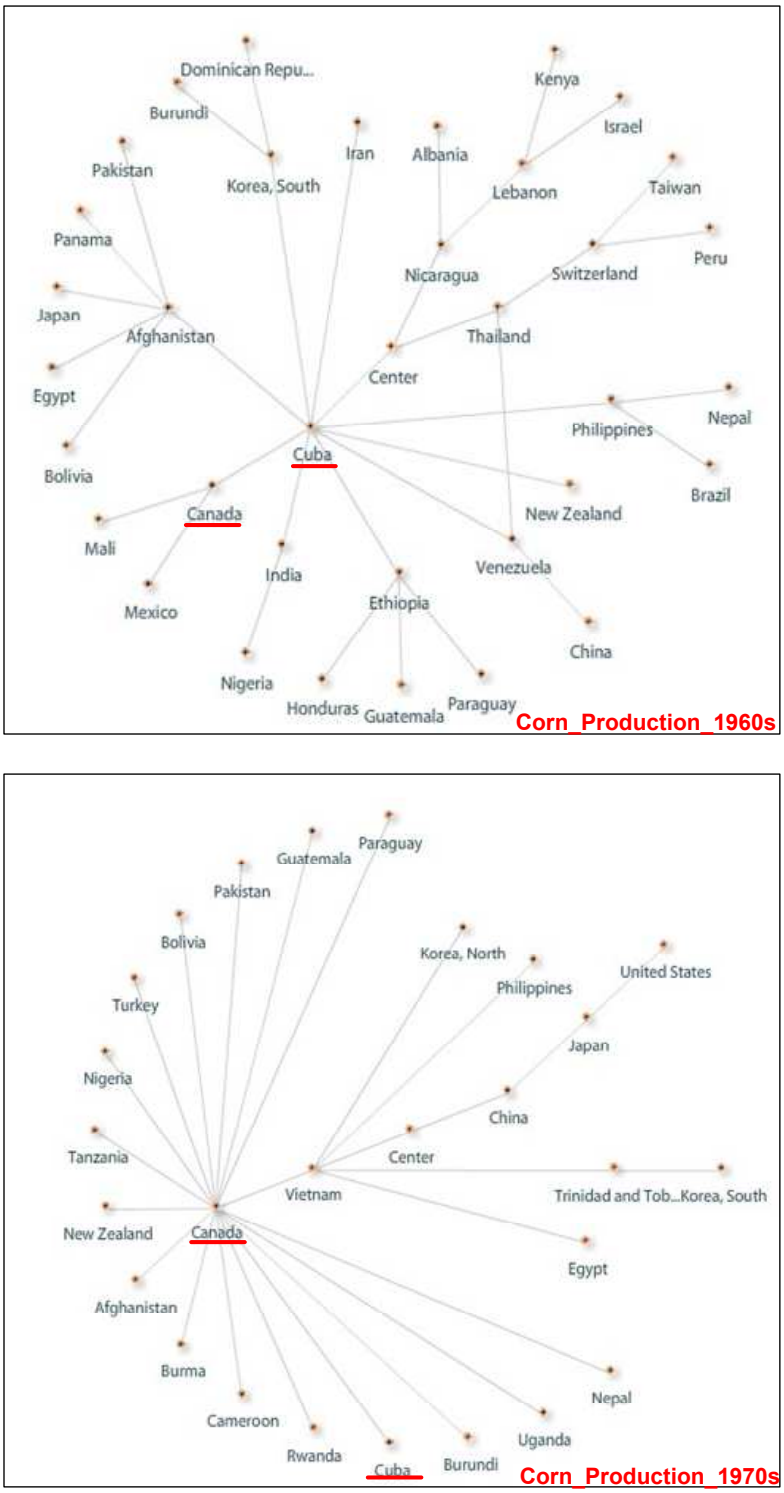


Figure 5.6: World corn production distribution from 1960s to 1970s

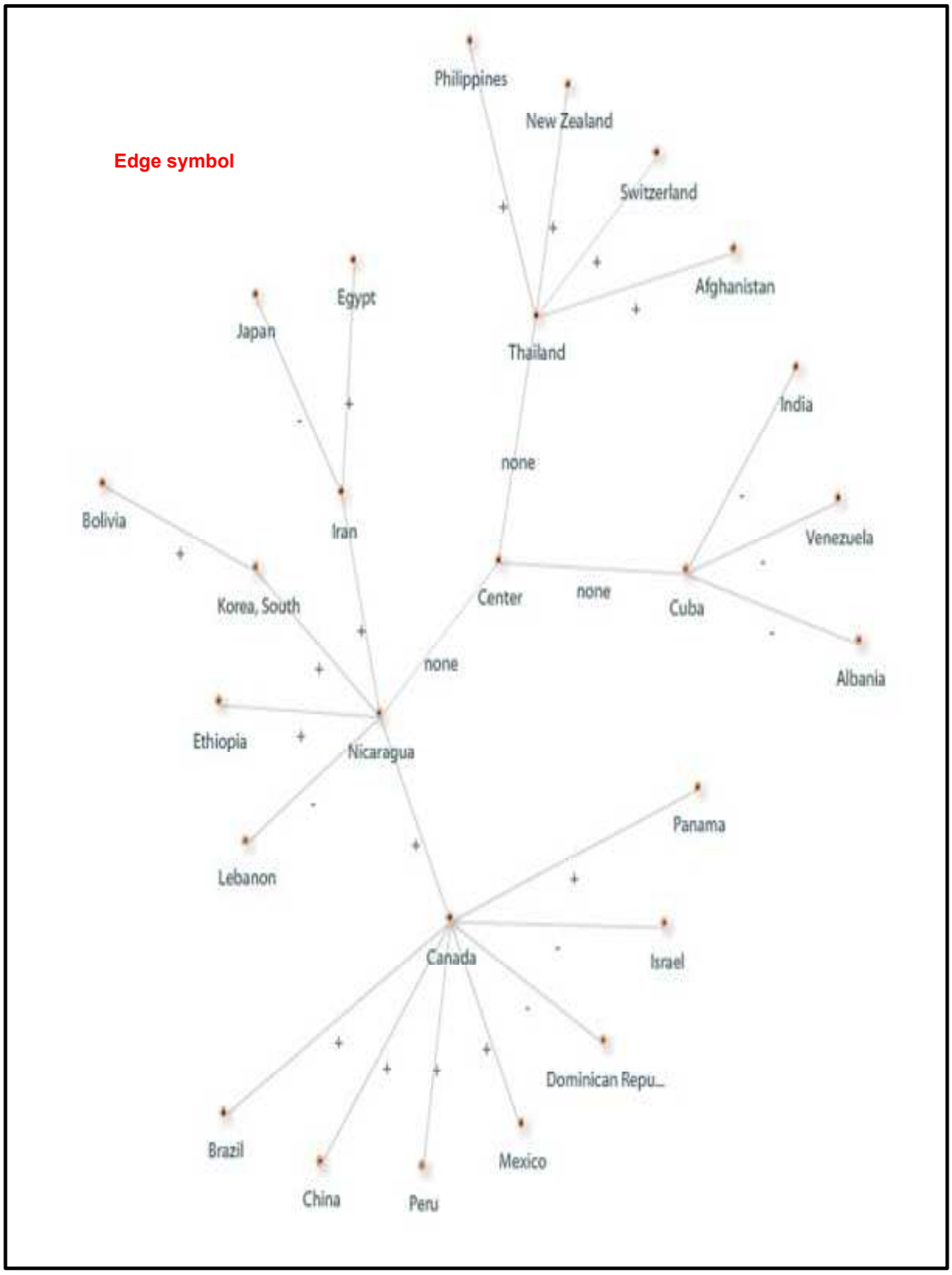
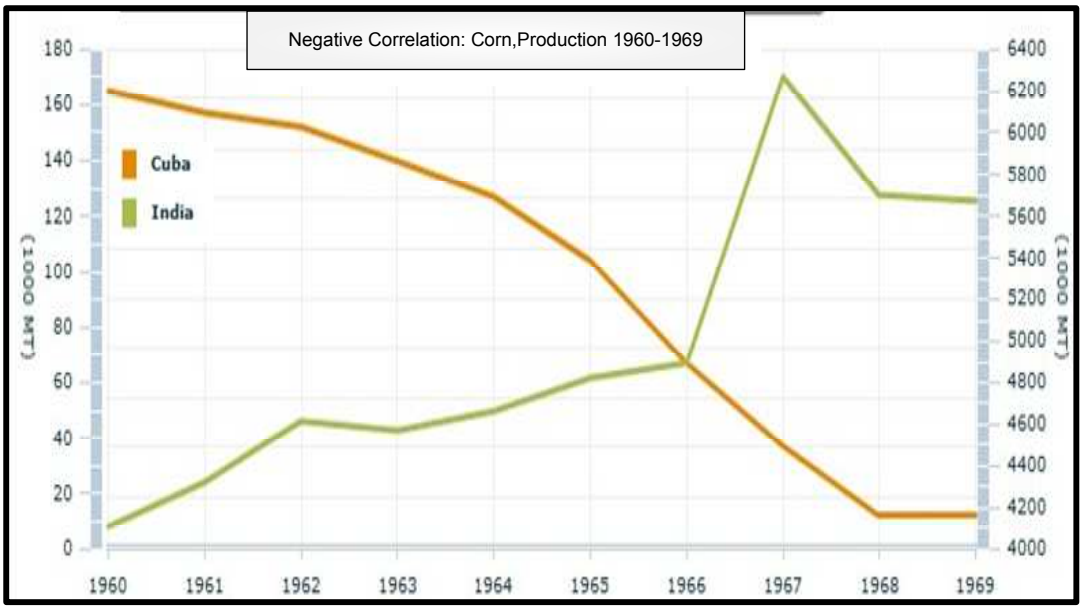


Figure 5.7: Edge label in a graph layout



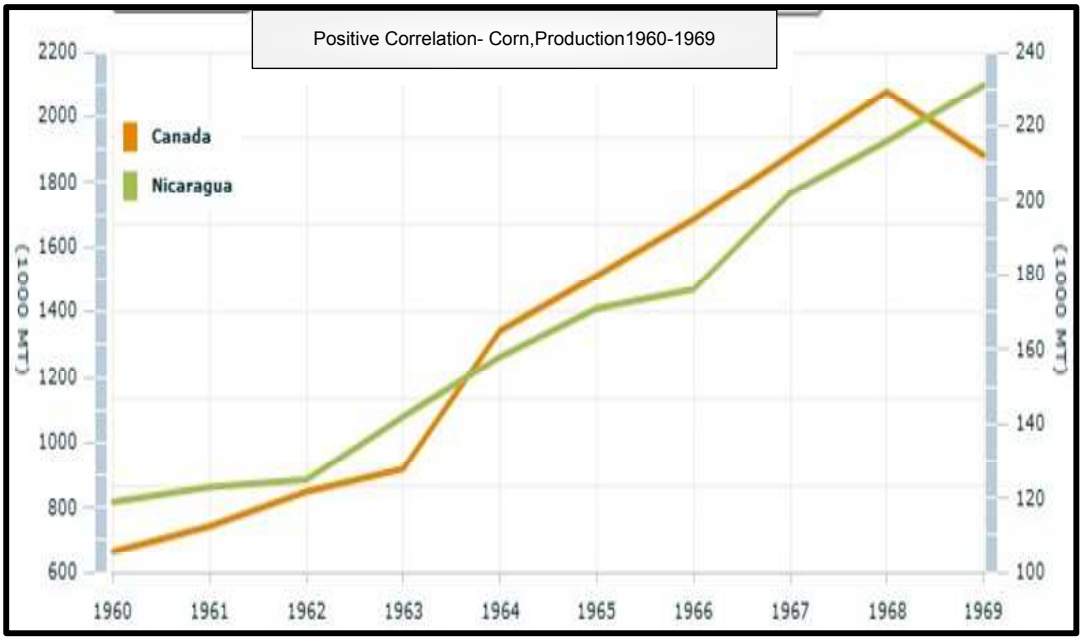


Figure 5.8: Positive correlation and negative correlation

sensitive way to display the relationships between nodes. (See Figure 5.9). According to the above description, we can make a conclusion that the unique features in our graph layout system bring a lot of advantages and benefits, and enhance the user experience. Additionally, the unique features also help clients understand our hierarchy algorithm. In the next section, we will discuss how our visualization system contributes to analyze the commodity market.

5.3 Meaningful Result in Energy Commodity Data

Our visualization system can be applied to different datasets, and it can generate some meaningful result and discover the hidden relationships in the data. In order to demonstrate our applications, the hierarchy algorithm is applied to many datasets. A persuasive example is when using our visualization system to analyze the price data in world crude oil market. Assuming that the price fluctuations of the crude oil would be high correlated within the OPEC or non-OPEC countries. According to world crude oil prices data source from EIA (the U.S. Energy Department's Energy Information Agency) website, we can see that EIA categorized world countries into three groups, including United States, OPEC countries, and non-OPEC countries(see Figure 5.10). In order to verify the assumption, we visualize the world countries distribution according to the price correlation in the world crude oil year by year. To be surprised, it was found that OPEC countries and non-OPEC countries are divided into two separate subtrees, and as a controller of world crude oil price, the United States is always displayed in an outstanding position of the visualized graph result (see Figure 5.11).

In addition, the changes in the graph layout results can help to explain the important events or transforms in the world crude oil market. For instance, the oil price history and analysis in the WTRG Economics [35] indicates that Russian oil production increases dominated non-OPEC production growth from 2000 forward and was responsible for most of the non-OPEC increase since the turn of the century. And meanwhile the fluctuations of crude oil price are influenced by the Russian breakout. Figure 5.12 shows the changes in the graph layout results in the crude oil price from 2000 to 2001.

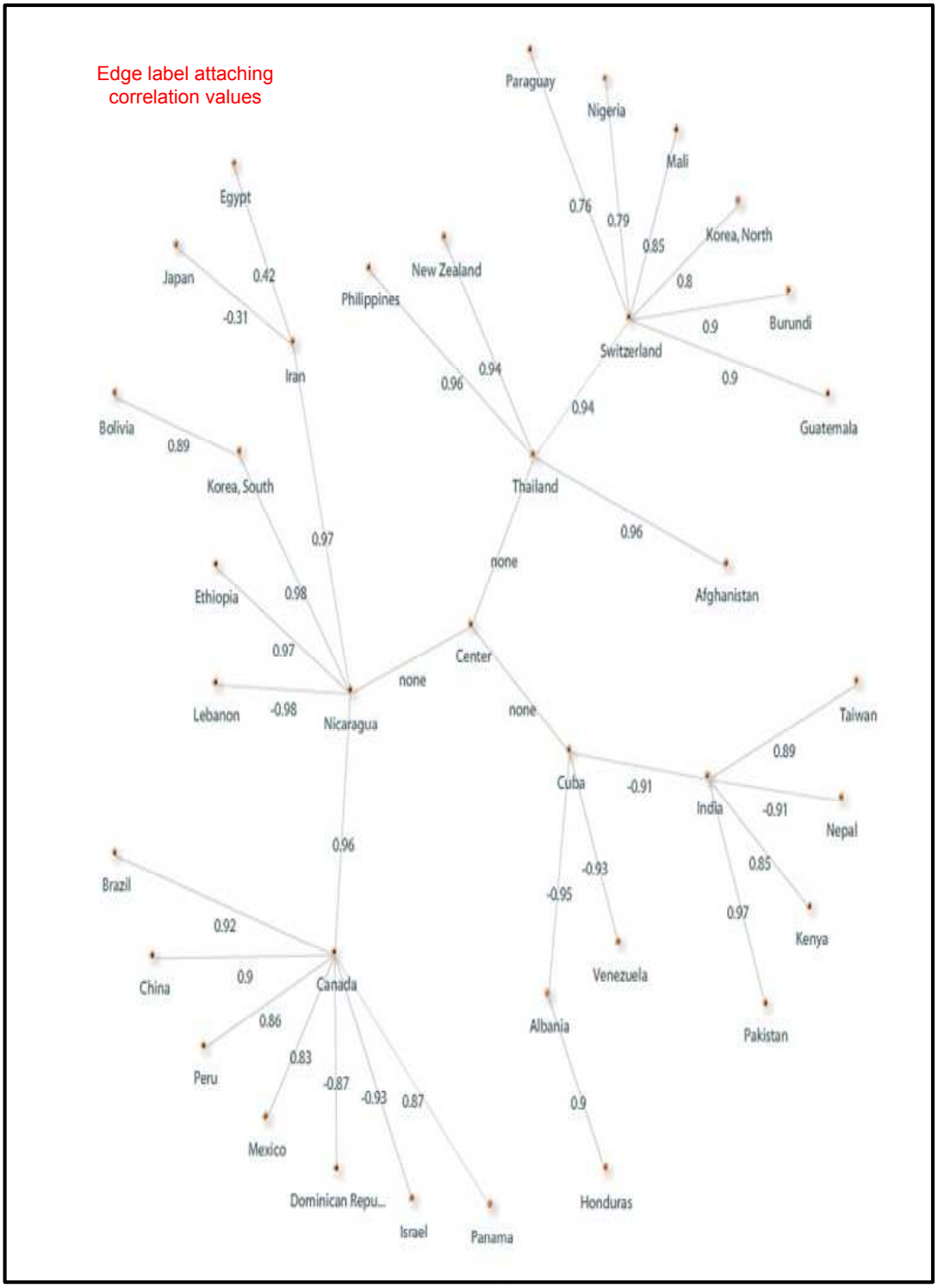


Figure 5.9: Edge label attaching correlation values

World Crude Oil Prices

(Dollars per Barrel)

Period: Weekly

Download Series History | Definitions, Sources & Notes

Crude Type	06/04/10	06/11/10	06/18/10	06/25/10	07/02/10	07/09/10	View History
Total World	70.97	70.68	73.85	75.09	73.31	71.11	1978-2010
United States	68.26	69.07	71.77	72.96	71.84	69.96	1978-2010
OPEC* Average	71.98	71.12	74.23	75.45	73.39	71.02	1978-2010
Abu Dhabi, Murban 39°	74.78	74.29	76.61	77.51	75.52	72.18	1978-2010
Algeria, Saharan Blend 44°	73.97	73.06	76.18	77.88	75.94	73.33	1978-2010
Angola, Cabinda 32°	73.73	72.64	75.63	77.13	74.89	72.32	1989-2010
Dubai, Fateh 32°	72.90	72.39	74.65	75.58	73.62	70.29	1978-2010
Ecuador, Oriente 30°	67.82	66.87	70.36	70.29	69.39	66.49	1978-2010
Iran, Heavy 30°	69.07	67.96	73.16	74.47	72.11	69.37	1978-2010
Iran, Light 34°	70.72	69.61	74.51	75.82	73.46	70.72	1978-2010
Iraq, Kirkuk 36°	73.13	72.25	75.11	76.38	73.97	71.75	1978-2010
Kuwait, Kuwait 31°	71.11	70.61	72.87	73.76	71.77	68.29	1978-2010
Libya, Es Sider 37°	74.22	73.09	75.94	77.35	74.76	72.36	1978-2010
Neutral Zone, Khafji 28°	70.56	69.79	72.84	74.09	72.10	70.46	1999-2010
Nigeria, Bonny Light 37°	75.61	74.56	77.56	79.03	76.84	74.34	1978-2010
Nigeria, Forcados 31°	75.56	74.60	77.63	79.10	76.89	74.34	1978-2010
Qatar, Dukhan 40°	74.27	73.74	76.09	77.00	75.07	71.78	1978-2010
Saudi Arabia, Arabian Heavy 27°	67.56	66.79	69.84	71.09	69.10	67.76	1978-2010
Saudi Arabia, Arabian Light 34°	70.56	69.79	72.84	74.09	72.10	70.46	1978-2010
Saudi Arabia, Arabian Medium 31°	68.66	67.89	70.94	72.19	70.20	68.71	1978-2010
Venezeula, Bachaquero 17°	NA	NA	NA	NA	NA	NA	1978-2010
Venezeula, Bachaquero 24°	NA	NA	NA	NA	NA	NA	1978-2010
Venezuela, Tia Juana Light 31°	72.25	71.02	74.23	75.15	73.97	71.86	1978-2010
Non-OPEC* Average	69.60	70.09	73.34	74.60	73.21	71.23	1978-2010
Australia, Gippsland 42°	74.48	74.02	77.34	78.94	76.91	73.90	1989-2010
Brunei, Seria Light 37°							1978-2003
Cameroon, Kole 34°	73.69	72.66	75.76	77.45	75.52	73.14	1989-2010
Canada, Canadian Par 40°	62.50	68.27	70.19	71.74	71.23	70.26	1993-2010
Canada, Heavy Hardisty 22°	57.73	63.10	66.31	67.61	67.13	66.28	2007-2010
Canada, Lloyd blend 22°							1989-2007
China, Daqing 33°	72.23	71.33	74.27	76.19	73.97	70.26	1978-2010
Colombia, Cano Limon 30°	75.00	73.89	77.13	77.38	76.50	73.76	1989-2010
Egypt, Suez Blend 33°	70.12	69.44	72.30	73.33	70.89	68.62	1978-2010
Gabon, Mandji 30°	NA	NA	NA	NA	NA	NA	1978-2010
Indonesia, Minas 34°	76.99	75.88	79.95	81.78	79.03	74.37	1978-2010
Malaysia, Tapis Blend 44°	75.13	74.75	78.00	79.87	77.96	75.01	1978-2010
Mexico, Isthmus 33°	72.14	70.91	74.12	75.04	73.86	71.75	1978-2010
Mexico, Maya 22°	64.19	63.25	66.08	66.78	65.83	64.22	1989-2010
Norway, Ekofisk Blend 42°	74.07	73.27	76.78	78.65	76.61	74.02	1978-2010
Oman, Oman Blend 34°	73.12	72.62	74.88	75.74	73.72	70.29	1978-2010
Russia, Urals 32°	72.23	71.16	74.87	75.97	74.46	72.53	1978-2010
United Kingdom, Brent Blend 38°	73.26	72.44	75.77	76.95	74.96	72.68	1989-2010

Figure 5.10: Category of world crude oil prices from EIA website

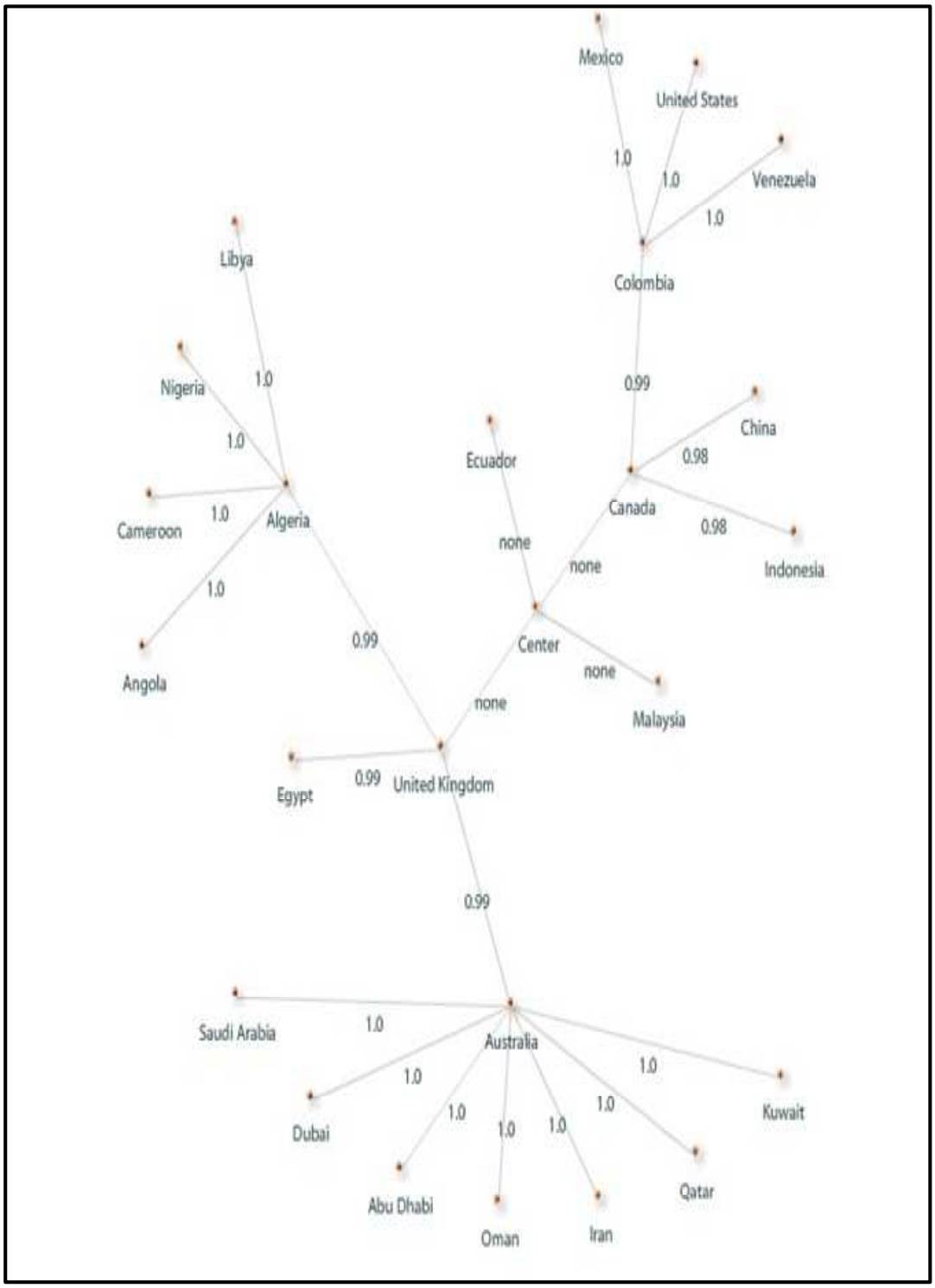


Figure 5.11: World crude oil prices distribution in year 1997

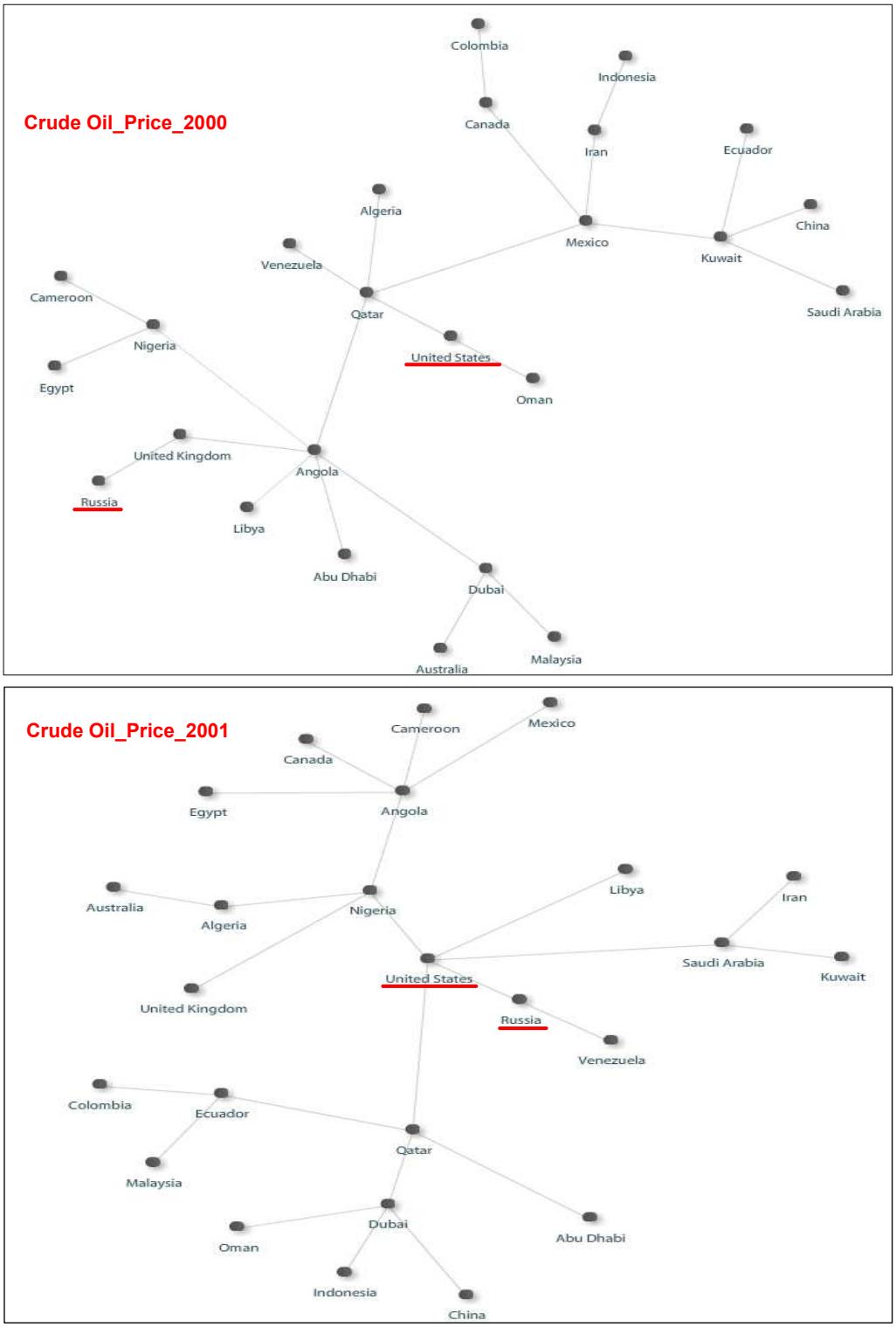


Figure 5.12: Graph layout changes from 2000 to 2001

Chapter 6

Conclusions And Further Work

In this thesis, we explore a novel hierarchical layout algorithm and build a robust multi-functional visualization online system. This novel hierarchical visualization algorithm provides a general stratification method to the time-series data in which the nodes degree distribution is fitted in a power-law. The core concept in our novel hierarchy visualization algorithm is to stratify nodes into different levels so that central and representative nodes in the graph are emphasized. In order to decide whether two nodes are connected by an edge, we demonstrate and explain the threshold selection method. In most cases, the selected threshold help to adjust the nodes degree distribution, so that the nodes degree is distributed according to a power-law. But, in a few cases, we also find that degree distribution cannot be perfectly fitted in a power-law distribution. In this case, the previous method of calculating the tree depth is not proper. The main reason is that this method evaluates the depth of a graph or tree depending on an exponent parameter in a power-law. To solve this problem, our hierarchy algorithm is first to sort nodes in a ranking formula, to place sorted nodes in levels and ensure that there is an equal or similar total degree in each level. And meanwhile, users can pre-set how many nodes they want to put in the first level. According to users' setting, we can calculate the depth of a graph using a new depth formula. Based on the graph layout analysis in financial data, we can conclude that our graph visualization algorithm generates a meaningful and understandable result and help users to evaluate and explore inherent relations in financial data.

Moreover, our visualization system implements many types of graph layout algorithms (such as hierarchical tree, radial tree, etc) and designs new features (such as degree zooming with new focus nodes, animated zooming, etc). Compared to the traditional graph layout system, our graph visualization system does not only integrate most traditional functions and features, but it also adds more interactive action with users and implements many kinds of popular layout algorithms. To enhance a successful user experience, we develop our visualization system in the Adobe Flex platform, which is a highly productive, free, open source framework for building expressive web applications that deploy consistently on all major browsers, desktop, and operating systems.

At last, we discover hidden relationships in the financial data using our hierarchical layout algorithm. And, the graph visualization system clearly displays the results in the multi-perspectives. When exploring the hidden relationships in the production of the world corn, we obviously find that the fluctuations of corn productions are highly correlated within the countries which are in the same continent. Besides, the relationships in the price fluctuations of world crude oil are discovered by visualizing a radial tree in this visualization system. The visualized results shows that the price fluctuations of world crude oil are quite similar within the OPEC or Non-OPEC group. Through viewing the visualization layout results in different time periods, we also can recognize the changes which indicates that specific events happened during the time periods. From 2000 to 2001, the center node in our hierarchical layout switches from United States to Russia. The oil price history and analysis in the WTRG Economics [35] indicates that Russian production increases dominated non-OPEC production growth from 2000 forward and was responsible for most of the non-OPEC increase since the turn of the century. And meanwhile the fluctuations of crude oil price are influenced by the Russian breakout.

Although our visualization system possesses a lot of advanced properties and the visualized results demonstrate many meaningful discoveries in the complex financial world, there are some issues that we need to solve and improve in the future. When building family relations in the hierarchy, this thesis only focused on finding the connection between nodes in

different levels but ignoring the relations in the same level. In addition, because our layout algorithms are applied on the short-length datasets which has a small size, we could not say that our visualization tool have good performance in calculation on a large size datasets. Although we use our visualization tool to create meaningful and understandable hierarchical layouts, the quality of the hierarchy is still very user-dependent. Moveover, there are some graph layout systems with the feature of $3D$ animation layouts. In that case, 3D visualization may generate many more meaningful results. How to develop and implement a novel 3D animation layout algorithm will be a future goal. The 3D layout will also can help to clearly visualize the relationships in the same level. Finally, this hierarchy algorithm is built in a tree structure only, other graph structures may generate more meaningful results. In future work, this algorithm will be adjusted to other hierarchy structures. Depending on the comparison between the tree hierarchy and others, the system can provide a variety of perspectives to users and help the clients understand the data.

All in all, in order to discover and visualize the hidden relationships in financial datasets, we develop a novel hierarchical layout algorithm and build a multi-functional visualization system. Based on our graph visualization system, we generate many meaningful results which verify that our layout algorithm and visualization tool will be helpful and useful in the analysis of financial data.

Bibliography

[1] I. Herman, G. Melancon, and M.S. Marshall, "Graph visualization and navigation in information visualization: A survey," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 6, no. 1, pp. 24–43, Jan-Mar 2000.

[2] D.M. Daniel W.J. Frank and R.F. Boisvert, *Zeta and Related Functions*, Cambridge University Press, 2010.

[3] R. Tamassia G.D. Battista, P. Eades and I.G. Tollis, "A heuristic for graph drawing," *Congressus Numerantium*, vol. 42, no. 42, pp. 149–160, 1984.

[4] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Inf. Process. Lett.*, vol. 31, no. 1, pp. 7–15, Apr 1989.

[5] T.M.J. Fruchterman and E.M. Reingold, "Graph drawing by force-directed placement," *Softw. Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, Nov 1991.

[6] G. Kumar and M. Garland, "Visual exploration of complex time-varying graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 805–812, Sept 2006.

[7] C. Walshaw, "A multilevel algorithm for force-directed graph drawing," in *GD '00: Proceedings of the 8th International Symposium on Graph Drawing*, London, UK, 2001, pp. 171–182, Springer-Verlag.

[8] D.S. Chan, C. Khim Shiong, C. Leckie, and A. Parhar, "Visualisation of power-law

network topologies," in *Networks, 2003. ICON2003. The 11th IEEE International Conference on*, Sept 2003, pp. 69–74.

[9] R. Tamassia G.D. Battista, P. Eades and I.G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Alan Apt, 1999.

[10] E.M. Reingold and J.S. Tilford, "Tidier drawings of trees," *Software Engineering, IEEE Transactions on*, vol. SE-7, no. 2, pp. 223–228, Mar 1981.

[11] J.Q. Walker, II, "A node-positioning algorithm for general trees," *Softw. Pract. Exper.*, vol. 20, no. 7, pp. 685–705, Jul 1990.

[12] J. Lamping, R. Rao, and P. Pirolli, "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies," in *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 1995, pp. 401–408, ACM Press/Addison-Wesley Publishing Co.

[13] L. Hatton, "Power-law distributions of component size in general software systems," *Software Engineering, IEEE Transactions on*, vol. 35, no. 4, pp. 566–572, Jul 2009.

[14] L.E. Diamond, M.E. Gaston, and M. Kraetzl, "An observation of power law distribution in dynamic networks," in *Information, Decision and Control, 2002. Final Program and Abstracts*, Nov 2002, pp. 101–105.

[15] T. Hirayama, S. Arakawa, K. Arai, and M. Murata, "On the packet delay distribution in power-law networks," in *Evolving Internet, 2009. INTERNET '09. First International Conference on*, Aug 2009, pp. 101–105.

[16] R. Wheeldon and S. Counsell, "Power law distributions in class relationships," in *Source Code Analysis and Manipulation, 2003. Proceedings. Third IEEE International Workshop on*, Sept 2003, pp. 45–54.

[17] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *SIGCOMM '99: Proceedings of the conference on Applications,*

technologies, architectures, and protocols for computer communication, New York, NY, USA, 1999, pp. 251–262, ACM.

[18] S. Butenkob V. Boginskiaand and P.M. Pardalos, "Statistical analysis of financial networks," *Computational Statistics & Data Analysis*, vol. 48, no. 2, pp. 431–443, Feb 2005.

[19] S. Butenkob V. Boginskiaand and P.M. Pardalos, *Innovations in Financial and Economic Networks*, chapter On Structural Properties of the Market Graph, pp. 29–45, Edward Elgar Publishing Limited, Inc., Jun 2003.

[20] E. Gansner, Y. Koren, and S. North, "Topological fisheye views for visualizing large graphs," in *INFOVIS '04: Proceedings of the IEEE Symposium on Information Visualization*, Washington, DC, USA, 2004, pp. 175–182, IEEE Computer Society.

[21] R. Albert, H. Jeong, and A. Barabasi, "Diameter of the world wide web," *Nature*, vol. 401, pp. 130–131, Sept 1999.

[22] M.S. Chen, J. Han, and P.S. Yu, "Data mining: an overview from a database perspective," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 8, no. 6, pp. 866–883, Dec 1996.

[23] R. Agrawal, C. Faloutsos, and A.N. Swami, "Efficient similarity search in sequence databases," in *FODO '93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, London, UK, 1993, pp. 69–84, Springer-Verlag.

[24] C.S. Li, P.S. Yu, and V. Castelli, "Hierarchyscan: a hierarchical similarity search algorithm for databases of long sequences," in *Data Engineering, 1996. Proceedings of the Twelfth International Conference on*, 26 1996, pp. 546–553.

[25] Wikipedia, "Correlation and dependence," `http://en.wikipedia.org/wiki/Correlation\_and\_dependence`.

[26] S. Dowdy and S. Wearden, *Statistics for Research*, chapter 9, p. 230, John Wiley, 1983.

[27] Jon M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, no. 5, pp. 604–632, Sept 1999.

[28] O. Astrachan, "Bubble sort: an archaeological algorithmic analysis," in *SIGCSE '03: Proceedings of the 34th SIGCSE technical symposium on Computer science education*, New York, NY, USA, 2003, pp. 1–5, ACM.

[29] H. M. Edwards, *Riemann's Zeta Function*, Academic Press, New York: Dover, 2001.

[30] BirdEye Group, "Birdeye information visualization and visual analytics library," `http://code.google.com/p/birdeye/`.

[31] G.J. Wills, "Nicheworks - interactive visualization of very large graphs," in *GD '97: Proceedings of the 5th International Symposium on Graph Drawing*, London, UK, 1997, pp. 403–414, Springer-Verlag.

[32] Y. Ka-Ping, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," aug 2001, pp. 43–50.

[33] EIA, "United states energy information administration," `http://www.eia.gov/`, jun 2009.

[34] USDA, "United states department of agriculture," `http://www.usda.gov/`, jun 2005.

[35] WTRG Economics, "Oil price history and analysis," `http://www.wtrg.com/prices.htm`.