Interactive Visualization of the Collaborative Research Network

by

Mohammad Alsukhni

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Applied Science (MASc)

in

Electrical and Computer Engineering

University of Ontario Institute of Technology

Oshawa, Ontario, Canada

(January, 2012)

Abstract

Social networks have been evolving over the past few years, leading to a rapid increase in the number and complexity of relationships among their entities. In this research, we focus on a large scale dataset known as the Digital Bibliography and Library Project or DBLP, which contains information on all publications that have been published in computer and information science related journals and conference proceedings. We model the DBLP dataset as a social network of research collaborations. DBLP is a structured and dynamic dataset stored in the XML file format; it contains over 850,000 authors and 2 million publications, and the resulting collaboration social network is a scale-free network. We define DBLP collaboration social network as a graph that consists of researchers as nodes and links representing the collaboration or co-authorship relationships among the researchers. In this work, we implement a data analysis algorithm called Multidimensional Scaling (MDS) to represent the degree of collaboration among the DBLP authors as Euclidean distances in 2-dimensional space in order to analyze, mine and understand the relational information in this large scale network in a visual way. MDS is a useful technique for data visualization and graph drawing methods, but it has high computational complexity for large scale graphs such as the DBLP graph. Therefore, we propose different solutions to overcome this problem, and improve the MDS performance. In addition, as the quality of the MDS result is measured by a metric known as the *stress* value, we use the steepest descent method to minimize the *stress* in an iterative process called stress optimization in order to generate the best geometric layout of the graph nodes in 2-dimensional space. We also propose a solution to further enhance the graph visualization by partitioning the graph into sub-graphs and using repelling forces among nodes within the same sub-graph. Finally, we developed a new visualization tool that can handle the large scale of the DBLP graph, and provides the user a number of significant features that allow them to explore, navigate and sift for information through the graph, such as graph scaling and graphical search functionality.

Acknowledgements

To my late brother,

There are no enough words in any dictionary I can use to express my gratitude to Dr. Ying Zhu for her kindness, efforts, dedications, encouragement, and standing with me in all my research phases from A to Z. I've learned a lot and I'm still learning from her: how to manage my research, how to face difficulties and how to overcome those through purely scientific ways.

I would like to thank Dr. Ken Pu for his valuable guidance, directions, and information he gave. He was abundantly helpful and supportive during my research.

Special thanks to ALL my friends here and back home for their support and encouragement, they know exactly what they mean to me.

To my father, and sisters, brother, and specially my mother who has not stopped worrying about me. This dissertation is dedicated to them.

Table of Contents

List of Figures

List of Tables

List of Acronyms

MDS Multidimensional scaling

DBLP Digital Bibliography and Library Project

IR Information Retrieval

DTD Data Type Definition

ERD Entity Relationship Diagram

CMDS Classical Multidimensional Scaling

SGD Steepest Gradient Descent

ID Identity

API Application Programming Interface

GUI Graphical User Interface

UOIT University of Ontario Institute of Technology

Chapter 1

Introduction

1.1 Introduction

Since the beginning of mankind on this earth, there has been development of social relationships among them, and with the settlement of human beings into different parts of the globe which led to the emerge of multiple cultures, and with the passage of time, human beings have developed different kind of customs, traditions and languages. Therefore the need of communication among them increased during this time.

Since the invention of the wheel, ways of traveling between human civilizations were developed, which led to the invention of cars, trains, ships and airplanes. And the communication media have been evolved from paper-based mail to electronic mail and from voice to video communication, and the advent of the Internet led to the spread of these new types of communications to grow exponentially among the people. Internet services became the favorite method to many users, and people started spending hours every day in using the Internet to accomplish their work, pay bills, and develop their social relationship through these services rather than using more traditional ways of communication.

Modern humans have transitioned to live in a virtual world, where they can build their friendships with the various races around the world. And because the complexity and diversity of these relationships grow over time, it has become imperative to study these relationships and find out how they have emerged and evolved, through building mathematical and statistical models.

People are connected to each other via the Web by sharing their information; people create their own social networks on the Web. These social networks are dynamic and continuously

developing and increasing in size and complexity, and it has become increasingly difficult to understand the how people are connected and how close they are to each other.

Social network data represents people's lives and their relationships on the Web; these data can be gathered, retrieved and then stored as relational datasets. One method can be used to study and analyze these relational datasets is to visualize them by using different types of graphs like where nodes represent individual or groups in a social network and links represent the relationships among them.

The purpose of collecting social network data is analyzing the relationships among the people, for instance, we may wish to find the similarities or dissimilarities between people based on the quality of their relationships. Due to the large size of the dataset, we believe that to facilitate an interactive interface for the user to explore and extract information from the data, the best way to use information or data visualization [1] One of the statistical techniques used to help in visualizing relational data is Multidimensional Scaling (MDS).

Multidimensional Scaling (MDS) is a set of techniques which are used to find similarities or dissimilarities in a relational data by transforming the information into distances in multidimensional space, where it can be used in visualization, pattern analysis, data preprocessing, and localization [2] MDS generates Perceptual Maps that present the relative positioning of all objects.

MDS analysis is based on comparing any objects with each other to find the distances between them. We must choose the objects which we want to evaluate, define the similarity measure, and decide whether the analysis should be applied on individual or group level. Also we must decide the type of data collected, whether they are quantitative (metric MDS) or qualitative (nonmetric MDS) data.

The relational social network dataset used in this research is called Digital Bibliography and Library Project (DBLP) which provides bibliographic information on major computer science journals and proceedings [3] DBLP contains more than 1.7 million publications, and more than 850 thousands authors; DBLP presents an excellent base to observe the evolution of the social network of the people who collaborate in computer science research areas.

In summary, social networks are created, evolved and develop by people creating relationships and communicating with each other on the Web or in the real life, e.g. firms in the market, people whose work in scientific research areas or even among people whose work in politics. Studying these social networks require creating mathematical models that can describe, and be used to analyze the relations among the objects belonging to these networks.

1.2 Motivation and Objective

DBLP contains all information about publications, journals, and venues that have been prepared and presented by people have been working in computer science research areas, DBLP presents a complex evolving scale-free co-authorship network, and considered one of the extensive databases those growths over time. DBLP is a mine of data which can be used in many study fields such as Information Retrieval (IR) [4] and Data Mining.

There are different ways to present the relations in social network data, such as graphs and matrices. Graphs can be used to make better understanding the patterns of ties among objects in social networks, on the other hand, visualization of the social network data helps in exploring and navigating of large scale social networks like DBLP, also helps in finding the connectivity of graph structure, providing visual search and analysis, visualizing communities in the network dataset and extract useful information mined from the networks such as extracting patterns, clusters, and trends.

MDS is in some cases high in computational complexity in terms of time and space [5]; as mentioned above, DBLP dataset contains more than 850 thousand authors or objects, therefore running MDS on this dataset will be costly in time and space required. It is in fact computationally infeasible to process proximity matrix which has the similarities or the distances measures between the objects in the network, proximity matrix is used to represent the relationships between the authors in DBLP dataset in term of distance. We propose several techniques to overcome the time and space problem in implementing MDS and still obtain the best geometric layout of the objects.

The main objectives of this research can be summarized in the following points:

- Due to the large scale of the DBLP dataset, implementing MDS will require a huge amount of memory and the optimization procedure which aims to minimize the difference between the proximities and the Euclidean distances among the authors which is known as *stress* , this process will require long time to converge to optimal stress, in this case the proximity matrix will have a size of $850{,}000 * 850{,}000$ which requires the proximity data to be represented using different structure that fits in memory.
- Visualization of the collaboration social network requires the stress to be minimized, and because MDS requires a long computational processing time. It is necessary to implement different structures to store the proximity data, to provide a fast access time to the data and a shorter convergence time for the algorithm.
- We wish to provide an interactive tool for the generated graph, for the user to freely analyze the characteristics and evolution of the collaboration social network, such as searching by publication year, by author affiliation, or by publication title.

- The DBLP dataset is dynamic and constantly updated, as a result, new publications or authors must be added and their relationship added as links in the social network; this means that the geometric location of any author may be changed due to the newly added links and/or author nodes, we will implement an object tracking functionally can be used to observe and track the evolution of the relationships between objects in the social network over time.

1.3 Methodology

In this research, we implement the 2D MDS technique to visualize the relationships found in any type of large scale social networks such as networks developed among people on the Web, between companies in business fields, between educational institutions … etc. Our research consists of the following steps: 1) Find a suitable relational dataset and convert it to a social network that has relationships among its entities. 2) Apply MDS technique in visualizing the similarities or dissimilarities between data objects in two-dimensional space. 3) Enhance the quality of the result and analyze the performance of MDS using different data structures. 4) Improve MDS runtime by using multithreading. 5) Finally build an interactive visualization tool to display the MDS results and provide the user with various functionalities that help the user in exploring and searching through the social network.

DBLP dataset is a real life large scale social network of people who have worked and published on all areas in and related to the computer science and information technology fields. DBLP data records are stored in XML format. Therefore, the first step we had to do was extracting this information and storing it in a relational database so we can access these records in a fast and easy way. The DBLP database was then used to create and construct the graph of the co-authorship social network. Authors are represented as points on the 2D plane and distances

between them indicates how close they are to each other, the shorter the distance between two authors the stronger the relationship they have. Our data has two kinds of distances, one comes from the similarities or strength of relationships among the nodes, and the other distance is the Euclidean distances between the points. The latter are randomly initialized using the uniform distribution.

Nonmetric MDS is an iterative algorithm that converges to an optimal solution. Nonmetric MDS requires a proximity matrix which contains all the dissimilarities between all pairs of nodes, and a two-dimensional matrix representing the initial configuration of the nodes. Unfortunately, it is computationally infeasible to find, store and process all dissimilarities between all node pairs, because of the huge scale of the social network and the huge size of the proximity matrix. The solution was to calculate dissimilarities for the nodes that have direct links between them and not for the other nodes i.e. those that have not co-authored any paper. Moreover to prevent nodes from being attracted close to the same location on the 2D plane we have divided the drawing graph boundaries into small sub-graphs or cells, and then added a repelling force between nodes that do not have direct links.

Nonmetric MDS iterations aim to minimize the node stress which represents a measure of the goodness of the solution. We chose to use the gradient descent algorithm to optimize the stress over the iterations. We have implemented different versions of nonmetric MDS by using three types of data structure to improve its performance. As well, we have implemented a multithreaded version of nonmetric MDS to expedite the stress optimization process.

The resulting visualization of the DBLP social network is a huge graph which it is impossible to understand or explore properly without tools. Therefore, we have developed a scalable visualized

graph with functionalities that the user can use to navigate through it, including the zooming functionality, and the ability to search graphically into the DBLP co-authorship social network.

1.4 Research Contributions

This research focus on implementing one of the most popular algorithms used in data visualization called Multidimensional Scaling (MDS), in order to visualize the relationships among the authors in DBLP dataset, and innovatively providing the user interactive capacities of searching and exploring the collaboration social network. In the following, we illustrate the main contributions of this research:

- DBLP is as a large scale and dynamic dataset of more than 850,000 authors and 2.7 million publications. Studying, analyzing, and understanding the relationships among the authors become increasingly difficult and complicated with the increase in number of authors, publications and venues. Therefore, data visualization becomes very important technique for facilitating the study of these large scale social networks.

- Visualizing the graph of the relationships between the entities of DBLP co-authorship social network requires a high quality result and fast and efficient techniques such as MDS which is considered a very efficient technique in visualizing the dissimilarities data. But MDS implementation becomes costly when we have a large scale and dynamic social network such as DBLP. Therefore, we propose cost-effective implementations of MDS algorithm in order to visualize this huge network.

- Searching and exploring the social network can be done by using different searching techniques and the results can be viewed as a text, tables, or charts. Human nature prefers viewing the search results graphically as rather than a text and tables. In this research, we

provide the user as interactive visualization tool for the DBLP co-authorship social network which enables him/her to search and explore through the social network and obtain the result as sub-graphs.

1.5 Thesis Organization

This thesis is organized in seven chapters. Chapter 1 is an introduction to the collaboration social network evolution, data visualization of the DBLP dataset, and the MDS algorithm.

Chapter 3 illustrates the DBLP dataset structure in its XML file, the parsing process of the XML file, and the relational database schema of DBLP dataset. Also in this chapter, we clarify how to enhance DBLP records by providing affiliation information of the authors, and how to process the tables to build the co-authorship social network graph structure.

In Chapter 4, we address several issues about using MDS algorithm in data visualization; also we present the difficulties of using MDS to visualize a large scale social network like DBLP co-authorship social network. As well, implementing and discussing the results of different approaches of nonmetric MDS, and finally, analyzing the result and the performance of the implemented nonmetric MDS algorithm.

Chapter 5 presents a multithreading technique used to build a multithreaded version of nonmetric MDS and discussing the improvements in the performance of the algorithm.

Chapter 6 gives details about all features in our developed visualization tool for DBLP co-authorship graph, which include graphical user interface (GUI) details, graph scaling, graph drawing options, and graphical search options. Finally, in chapter 7 we briefly summarize what have been done in this research, and discuss some future works and improvement can be applied to this research.

Chapter 2

Literature Review

2.1 Background

The DBLP graph consists of nodes with links among them, each link between two authors represents that at least one publication that has been coauthored by them, and the weight of the link is essentially inverse of the number of shared publications that have been co-authored by them. Our co-authorship graph consists of 854332 nodes and 2,793,603 links.

2.2 Evolution of DBLP and Social Networks

Digital Bibliography and Library Project or DBLP has been evolved from a small experimental Web server in 1993 to a computer science community which consists of millions of bibliographic records; these records represent bibliographic details of variety types of articles which have been published in journals, conferences, or Web pages. The DBLP records are stored in huge XML file[1]. These records are freely available for researcher to test their algorithms. In addition, all XML records contain the following two elements: article's key which is considered as a unique key for each record, and the date of the last modification. Other attributes mostly found in each record are: author/s name, title of the article, pages, year, and cross reference [6].

DBLP is not well designed, because it lacks citation information [10] and it doesn't support person ID to identify the author, which makes a conflict with the naming conventions of the author, and because DBLP has around 700,000 different names, we may locate the same author in different naming styles. Therefore the problem of Synonyms and homonyms will be more complicated in the near future.

[1] Size of DBLP XML file is 894MB, 2011/10

DBLP represents a co-authorship social network where the person plays the main rule in this network. Co-authorship social networks evolve through increasing the number of persons/authors and the complexity of theirs relationships with each other. There are three different approaches to study the characteristics of a social network: empirical measurement to find out topological measures of the network characteristic of any time, analytical measures that uncover the network's time evolution, and numerical simulations used to find out the network behavior [11]

A.L Barabasi *et al.* [11] studies the evolution of the co-authorship social networks related to all publications in the field of mathematic (M) and neuro-science (NS) on the period from 1991 to 1998. Figure 2.1 shows the extensive difference of the authors and the publications between DBLP, NS and M datasets.

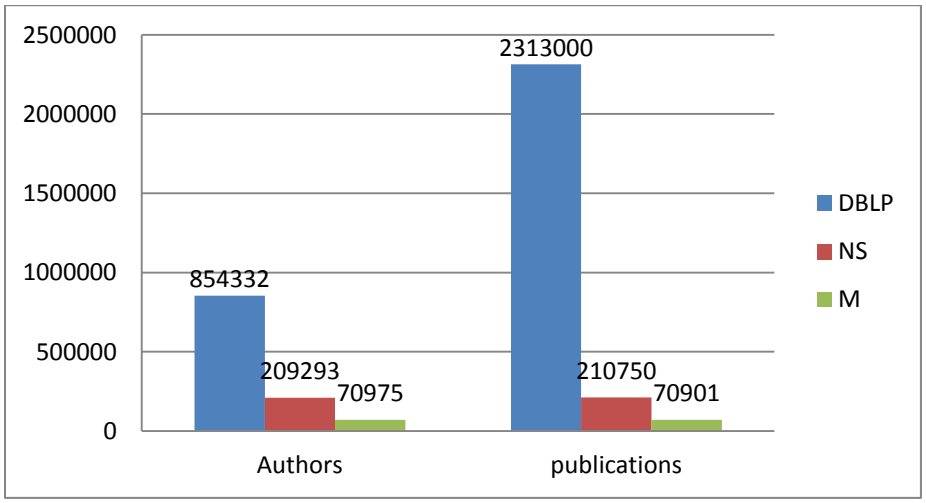


Figure 2.1 Number of authors and publications of DBLP, NS and M datasets

M and NS social networks are free-scale networks [12] because their degree distribution follows a power-law. See Figure 2.2 which shows the degree distribution for three graphs: M, NS and DBLP, which prove that our graph is a scale-free network.

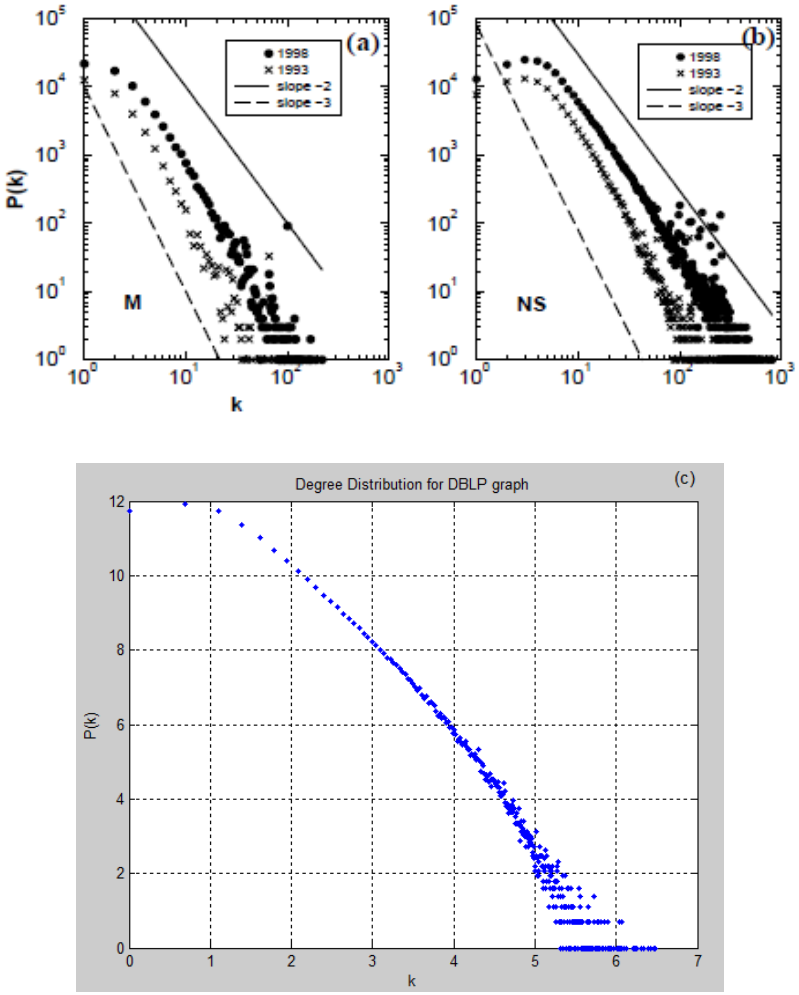


Figure 2.2 Degree distribution for (a) M, (b) NS [11] and (c) DBLP graph

2.3 Multidimensional Scaling

Multidimensional scaling (MDS) is a data analysis technique which is used to view the data into multidimensional space where distances represent the similarities or dissimilarities of any pairs of objects. The input data of MDS algorithm is proximity matrix and the output is a new configuration of the objects in *n-D* space. MDS can be classified into different types which

depend on the characteristics of MDS models, and the number of proximity matrices. Such these types are:

- Classical MDS: the proximity matrix represents the dissimilarities between any pair of object as a metric prosperities, and the result of this is an coordinate matrix with configuration that preserve the ratio between the proximities as good as possible.
- Metric MDS: transform a distance matrix into a new coordinates in p-dimensions so that the Euclidean distances between the object in the matrix as good as possible to the original distance.
- Non-metric MDS: the proximity matrix represents the dissimilates between the objects and it object to find a new configuration of the nodes which is represented as Euclidean distances so that it reflects the order of the proximities as good as possible.

In this research we focus on non-metric MDS because our data has dissimilarities and we have to find a new configuration of this data in 2-dimensional space. Objects are represented as points in 2D space, and Euclidean distance d_{rs} between any two pair of points as close as to the dissimilarities δ_{rs}. The accuracy of the distance between d_{rs} and δ_{rs} is calculated by the stress S which is expressed in equation (2.1) [13].

$$S = \sqrt{\frac{\sum_{r,s}\left(d_{rs} - \hat{d}_{rs}\right)^2}{\sum_{rs} d_{rs}^2}} \tag{2.1}$$

Where $\hat{d}_{rs}$ is monotone least-squares regression of d_{rs} and δ_{rs}.

The Euclidean distance between any two objects r and s is given by the following equation

$$d_{rs}(X) = \left(\sum_{l=1}^{p} (x_{rl} - x_{sl})^2 \right)^{\frac{1}{2}} \tag{2.2}$$

Where X is the coordinate matrix of the objects, and p is the dimensionality of the solution.

MDS is an iterative algorithm aims to find a matrix X so that the stress value is minimized [14]. The value of the stress depends on the size of the dataset. Mainly stress has a higher value with larger dataset [15].

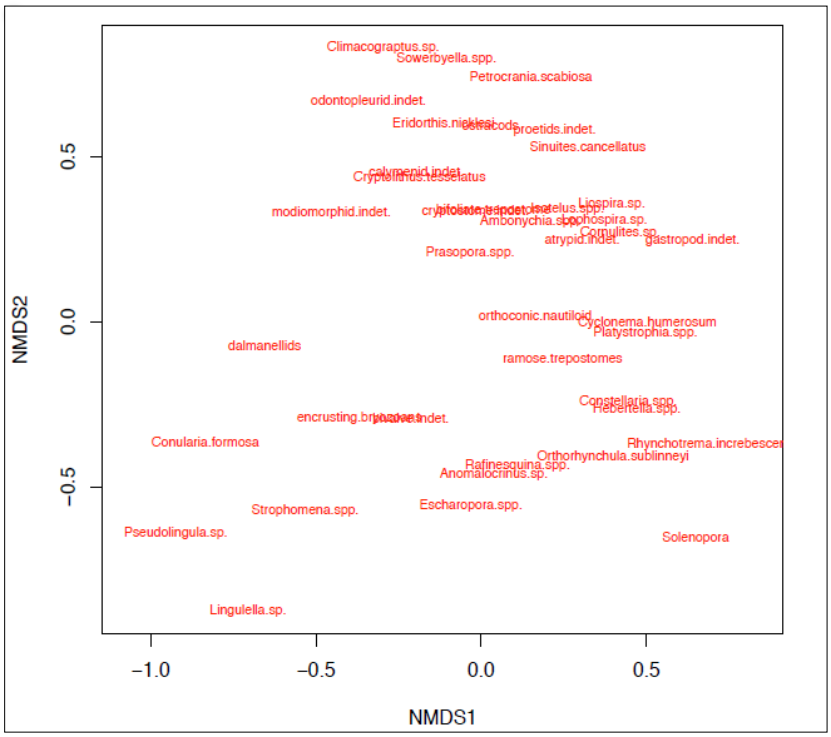


Figure 2.3 MDS solution in 2D of sample of data [15]

2.4 Graphs

Daniel Tunkelang [16] proposes different improvements for general graph drawing algorithms in terms of speed, quality of the drawing, and flexibility. In his work, Graph G is defines as a set of objects with relationships among them. Objects are defined as vertices or nodes V and

relationships are indicated as edges or links E. The process of transforming a graph from a set of data to visualized version is defined as drawing, see Figure 2.4. Also there are verities of drawing conventions can be used to draw a graph such as: poly-line, straight-line, orthogonal, planar, and upward drawings.

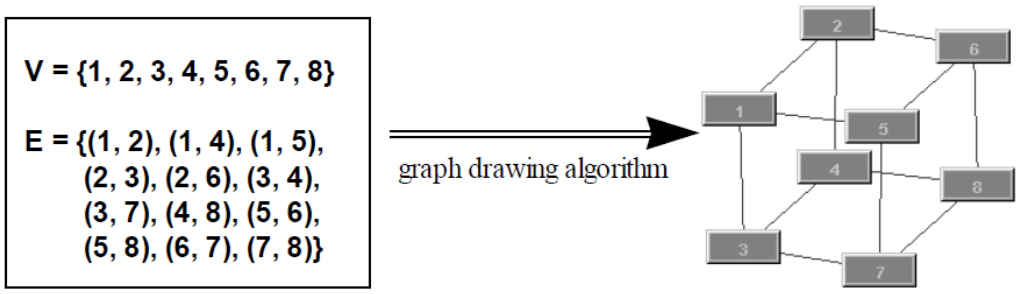


Figure 2.4 Transforming graph G structure into drawing [16]

Many approaches have been proposed in order to solve graph drawing problems, one of these approaches is known as Force-Directed Approach where a group of different kind of forces applied on the graph vertices, as a result their final configuration is optimum with a net force equal to zero on each vertex. This approach takes two components into consideration: energy and optimization algorithm. In the following we discuss the verities of force-directed approaches:

- The Spring Embedder Model: in this approach edges are represented as springs applying forces on their endpoints, and vertices are represented as positive electrical charges. This approach uses Eadses's optimization algorithm that applies a specific number of steepest descent iterations [17]
    - Complexity: for graph with n vertices and m edges, the computational cost is $O(n^2)$ plus the computation cost comes from steepest descent iterations. Therefore the overall performance is $O(n^3)$.

- Advantages: generates a good graph layout for small graphs.
- Disadvantages: bad graph layout for large and dense graphs with many edges.

- Kamada and Kawai's Approach: this approach eliminates the electrical charges in spring embedder model and replaces them with spring forces among each pair of vertices. "*For each pair of vertices, Kamada and Kawai make the spring's rest length proportional to the shortest path in the graph connecting the two vertices associated with the spring, and the spring's stiffness inversely proportional to its rest length*" [18] . The optimization algorithm used in this approach moves one vertex per iteration to minimize the locally minimal energy of the vertex.
  - Complexity: this approach requires $O(n)$ time per iteration to move only one vertex, also calculating all shortest path in the graph as preprocessing step, therefore the computational cost of this approach is $O(n^3)$.
  - Advantages: generates better graph layout than Spring Embedder model.
  - Disadvantages: costly in term of time because it requires $O(n^3)$, and space because it requires $O(n^2)$.
- Fuchterman and Reindolg's Approach: this approach modifies the spring embedder model in order to enhance its computational performance by "*applying spring forces that attract the endpoint of the edged in proportion to the square of the distance between them*" [19].
  - Complexity: the same computational complexity explained in spring embedder model.
  - Advantages: simple and faster than spring embedder model, because it determines the degree of their movement according to a "cooling schedule", and

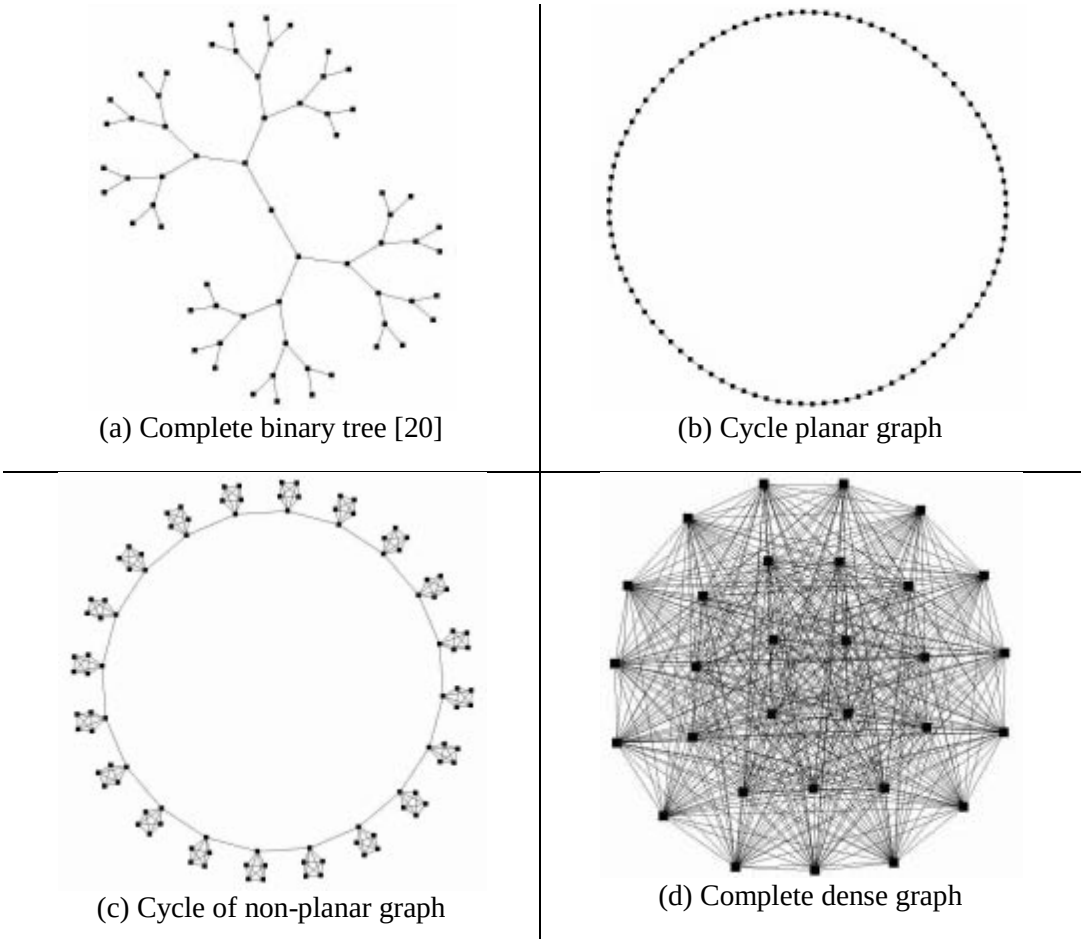
simulated annealing method is used so that it limits that distance that vertex can move which decreases the number of iterations performed.

- Disadvantages: it has a fixed number of iterations. As well it is not suitable for large size graphs.

Optimization in the graph drawing aims to get the optimal configuration of the vertices on the graph layout. Optimization procedures are calculated iteratively by performing gradient computations. The graph drawing is improved during the iterations by deciding two key factors: the moving direction and the step size.

Steepest Descent uses negative gradient method as a direction, thus vertices on the graph move in the direction of the force. The main advantage of this method is simplicity, but it always converges to a local minimal.

Figure 2.5 shows a variety of drawings produced from the above approaches on a graph such these drawings are: trees, planar, non-planar, and dense graphs.



(a) Complete binary tree [20]

(b) Cycle planar graph

(c) Cycle of non-planar graph

(d) Complete dense graph

Figure 2.5 Variety of graph drawings [16]

2.5 Data Visualization of DBLP

José F. Rodrigues *et al.* [21] addresses the challenges that may be encountered in visualizing large scale graphs like the DBLP co-authorship social network. They have developed a multi-resolution graph exploration visualization tool called GMine that can handle these large scale graphs. This solution is based on partitioning the graph into hierarchy of communities or sub-graphs, so each sub-graph has a minimum number of edges to other sub-graph. The resulting of graph partitioning forms a R-tree graph -tree data structure used here to represent a graph in order to perform quick indexing- which is shown in Figure 2.6.

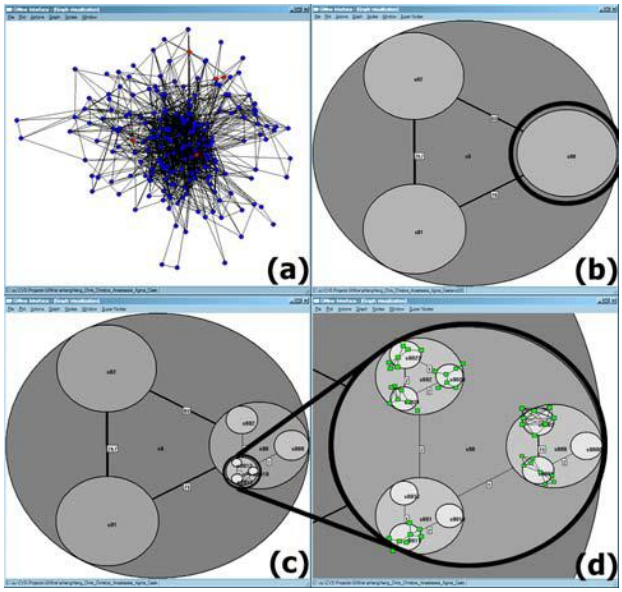


Figure 2.6 (a) A 200 nodes sub-graph extracted from DBLP. (b) Tree partition of the same graph. (c) One level down the hierarchy and we have three other communities inside the community highlighted in (b). (d) Zoom in the community highlighted in (c) and another level down the hierarchy [21]

This tool gives the user the ability to navigate and explore the graph in hierarchical approach, and speeds up the process of exploration of large graphs. According to this study the graph layout will change because authors in DBLP dataset may start to belong to different communities during the partitioning process that may generate an incorrect graph layout, especially for strongly related authors those may become far away from each other in the graph layout.

Stefan Klink *et al.* [22] have developed a new data browser and co-authorship graph visualize for DBLP dataset called DBL-Browser, this tool provides the user by textual and visual browsing features such as charts, histograms, and graphs like co-authorship and conference/journal relationship graphs. In this study, visualization of co-authorship graph shows the similarities between coauthors according to distance and the thickness of the link connected to the actual

author. Unfortunately, this study is limited to a small part of co-authorship graphs, and it is not efficient for large and cluttered co-authorship graphs.

Bradley Huffaker *et al.* [27] developed a general purpose visualization tool called Otter for wide verity of Internet data. Otter can handle any formatted dataset consisting of nodes, links or paths. It can be used to visualize a large datasets, support the user with geographical or topological placement, and provide the user with zoom, focus and manually tweak the graph layout functionalities. Otter uses placements algorithm to draw the graph layout which has two stages: "*positioning the subset of root nodes and subsequent positioning of other nodes relative to these root nodes*". Otter has two options to root node placement: Circular where nodes are places along the circumference of a circle, and coordinate-base where the (X, Y) coordinates are provided within the input file for all nodes as a geographical locations. For non-root nodes, they are placed according to their root nodes (parent nodes) by using breadth-first scan from the roots provided by the following heuristic information: 1) nodes with more children must placed away so they don't overlap with other's children. 2) children with more children must placed further away to prevent any overlap with other grand children. See Figure 2.7 which illustrates circular layout for a graph using circular placement algorithm.

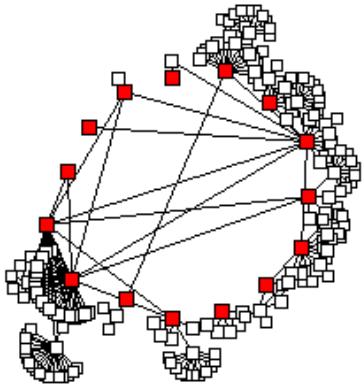


Figure 2.7 Circular Layout for Circular placement algorithm

Figure 2.8 shows a how a manually tuning of a graph works in (a) and (b), and it shows how zooming functionality helps the user to scale the graph in (c) and (d).

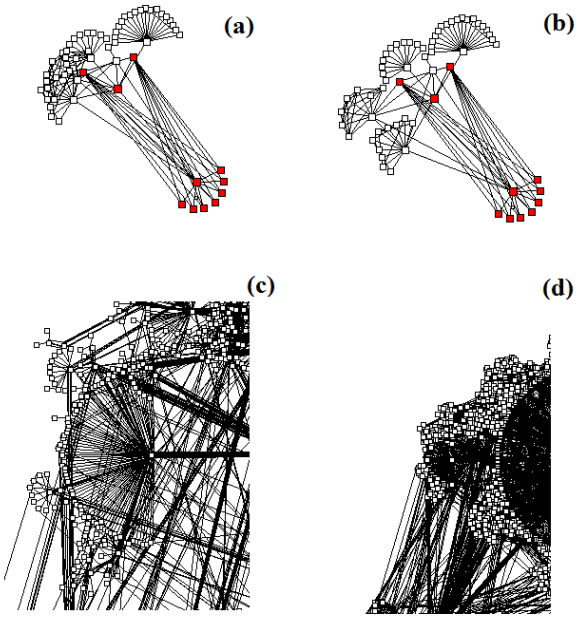


Figure 2.8 (a) Before manual placement. (b) After manual placement. (c) Otter zoomed in. (d) Otter zoomed out

Otter performance is limited to the graph size. Otter can handle datasets with 200-700 nodes. for large scale datasets involve tens of thousands of nodes, Otter placement algorithm is unable to generate a high quality graph layout, because many nodes mapped close to each other which obscuring the view from the user. Therefore, all algorithm that had been used were unable to view the graph layout of such large datasets. The only solution they had is to use pre-computed (X, Y) layout coordinates, so that the nodes will be placed directly without the need of exhaustive graph layout process. See Figure 2.9 which shows a graph layout consists of 35,000 nodes, the placement algorithm took 24 hours to pre-compute the layout coordinates and a minutes to draw the result.

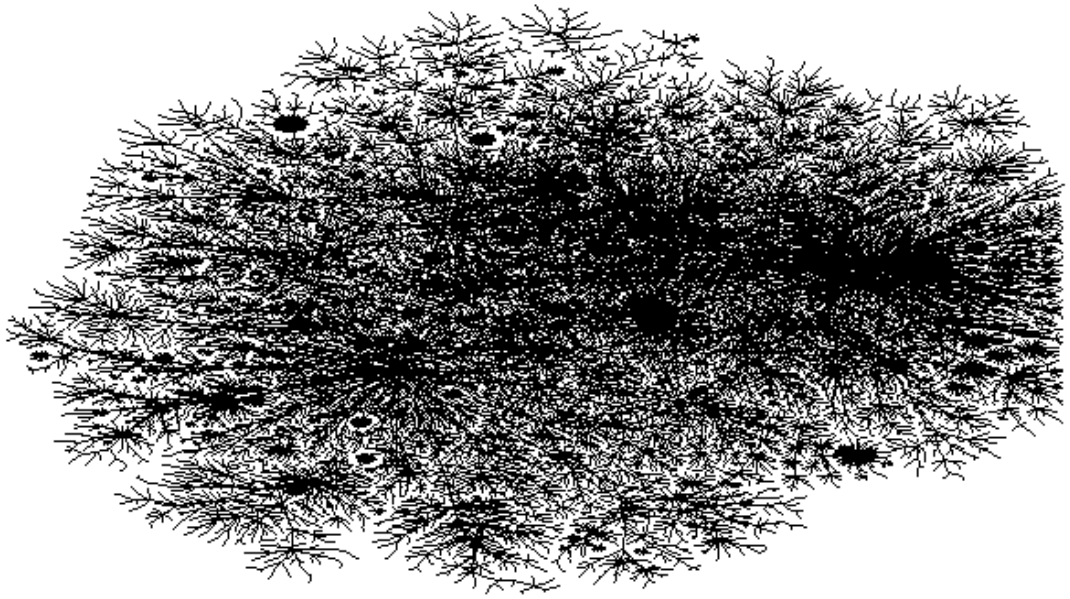


Figure 2.9 Otter result for pre-computed graph layout

Chapter 3

DBLP Dataset

3.1 DBLP Dataset Structure

Digital Bibliography and Library Project or DBLP has bibliographic information on published journal, articles, and conferences proceedings in the field of computer and information science [3] DBLP data records are dynamic and updated over time. The dataset is stored in a large XML file, we consider DBLP in the light of large scale collaboration as social network which can be explored through visualization and graphs such as individual publications graphs, or coauthor graphs which are of interest representation in this research [6].

DBLP records are stored in XML file which has the following format:

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
record 1
...
record n
</dblp>

The DBLP XML file comes with Data Type Definition (DTD) file which is important for reading and parsing the records, DBLP records in the XML file start with the root element <dblp>. See Figure 3.1 **Error! Reference source not found.**to understand the structure of DTD file, DBLP dataset records have eight different elements of publications and for each main element there are sub elements which provide information regards to any publication such as authors, year, title, journal, pages … etc. The next paragraph will explain the major elements of DBLP dataset records.

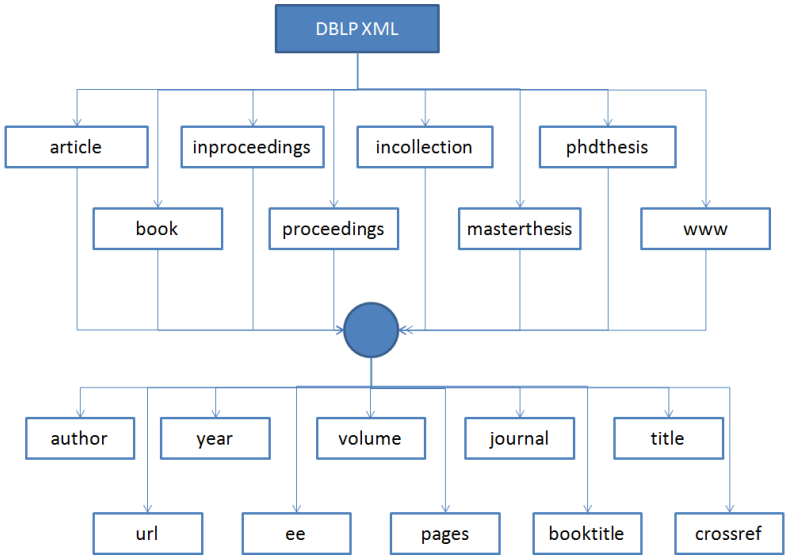


Figure 3.1 DTD elements used to present the DBLP bibliographic records

As mentioned above, XML records have different publication types called elements such as articles, inproceedings, phdthesis, books … etc. Each one of these elements has sub elements shown in the following snapshot of DBLP records.

```
<incollection mdate="2002-01-03" key="books/acm/kim95/ChristodoulakisK95">
        <author>Stavros Christodoulakis</author>
        <author>Leonidas Koveos</author>
        <title>Multimedia Information Systems: Issues and Approaches.</title>
        <pages>318-337</pages>
        <year>1995</year>
        <booktitle>Modern Database Systems</booktitle>
        <url>db/books/collections/kim95.html#ChristodoulakisK95</url>
</incollection>
```

In the following, we provide a brief description of some major elements frequently found in the DBLP records:

- **Author:** most of the publications have at least one author, if the publication's author is unknown, then this element doesn't have the author's tag. Some publications may have more than one author. Also author names can be read in different ways according to the

naming traditions for each country or region. Unfortunately, DBLP records don't have an ID that can be used to identify the authors. Therefore, the DBLP treats the same author who has different name conventions as individual authors.

- **Title:** represents the title of a publication. Furthermore, all publications in DBLP dataset must have this element in its records.
- **Pages:** represents the page number or range of pages of a publication, pages tag has the following format *<pages>from-to</pages>*, if a publication has one page only, then the page number will be without hyphen.
- **Year:** represents the year when the conference took place or when the publication was published, also this element must have four digit numbers in all DBLP records that have this element.

3.2 XML Parsing

The DBLP records - which are stored in XML format - represent a sequence of historical information, where each period of time new records added to this file. Moreover, there is no relational information between these records. Therefore, the process of parsing and preparing must be taken before start working on these records. The main purpose of parsing the XML records is to extract some useful information which can help us to build a relational database that provides both a relational data to build co-authorship social network and faster access to the data than XML structure. That because, the process of XML parsing is considered highly expensive operation in term of CPU utilization and larger memory consumption which decreases the query performance.

3.2.1 SAX and DOM Parsers

There are variant XML parsing models used to read and manipulate XML files. Such these parsers are SAX, DOM, StAX, and VTD parser. The parsing process goes through three-step processing (Character conversion, Lexical Analysis and Syntactic Analysis). These steps are considered the most expansive stages in XML processing [7].

The process of choosing between these parsers depends on the application requirements. Based on that, the suitable parser for DBLP dataset must meet the following requirements:

- Parsing XML records element by element.
- Sequential access to the records.
- Can deal with large XML files.
- Doesn't reserve a large space of the memory.

We have prepared a comparison between SAX and DOM parsers to choose the suitable one for our DBLP parser application. Table 3.1 shows a comparison between DOM and SAX parsers, and why SAX parser is considered the best parser that meets our application requirements.

Table 3.1 DOM vs. SAX parser

Key to compare	DOM	SAX	Application needs
Output	Tree object	Events based parser	Both are suitable
Parsing speed for small files	High	slow	Both are suitable
Parsing speed for large files	slow	High	SAX is better because DBLP dataset has a huge size > 800 MB

Access to the data	Fast (random access to the document)	Slow (sequential access)	SAX is better because we need to pass through the file only once.
Memory usage	Store the XML document into memory.	Doesn't store the XML document into the memory	Sax is better because we have limited memory resources
File Size	Recommended for Small size documents	Support any document size.	Sax is better because of the DBLP XML file size.

SAX parser meets our application requirements in term of parsing process stages, memory usage, and file size. SAX parser is events based, supports simple form of data, doesn't require a large memory space, and more efficient in dealing with huge size documents. The next section explains the process of storing the parsing outcomes in to a relational database.

3.2.2 DBLP Relational Database Schema

The main objectives of storing DBLP records in a relational database are to construct the DBLP co-authorship social network, and a relational database provides easy and fast access to the required data. Figure 3.2 illustrates the Entity Relationship Diagram (ERD) of the relational database schema for the DBLP database.

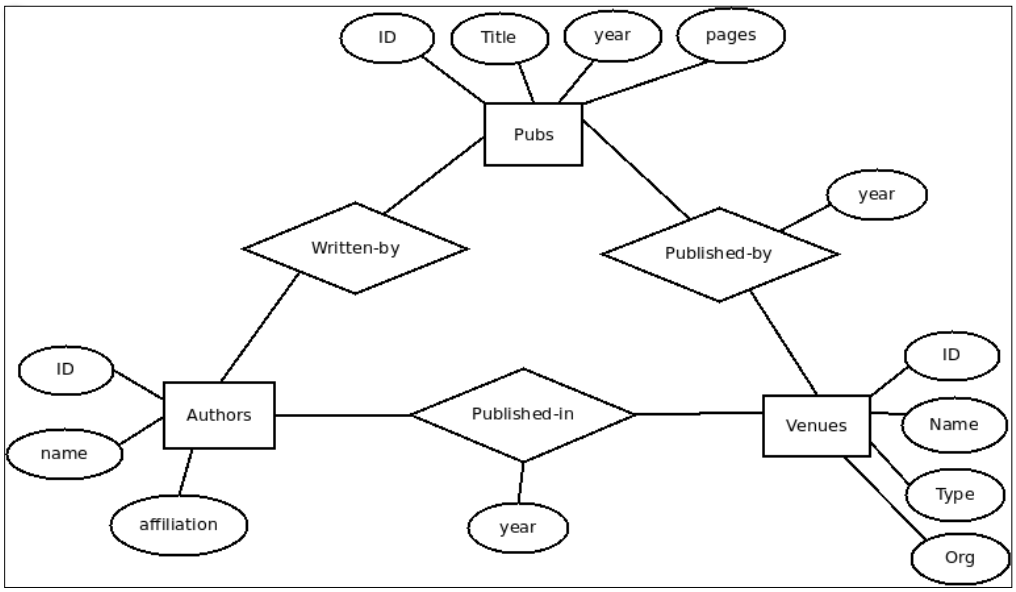


Figure 3.2 ER diagram of the relational schema for DBLP database

DBLP database consists of the following tables:

- Authors table:
    - ID: each author has a unique ID which considered as a primary key.
    - Name: this attribute contains the names of the authors and the coauthors whose have publications in the DBLP records.
    - Affiliation: affiliation information is not provided in DBLP dataset, so that a new Web IR module has been built to retrieve this information from the Web.
- Venues table:
    - ID: each venue has a unique ID.
    - Name: this attribute represents the title of the conference where the publications have been published.
    - Type: refers to the main topic of the conference.
    - Org: the organization that held conference in.

- Pubs table:
    - ID: each publication has a unique ID.
    - Title: the title of the publication.
    - Year: the year when the publication was published.
    - Pages: the pages where the publication were taken from.
- Written-by table: a relational table has two primary keys: author_id and pub_id. Where author_id represents an author ID and pub_id represents a publication ID.
- Published-by table: a relational table has two primary keys: pub_id and venue_id. Where venue_id represents a venue ID.
- Published-in table: a relational table has two primary keys: author_id and venue_id.

We implemented and developed a SAX parser to fill the DBLP database tables with the required data and build the relational tables. See Table 3.2 which presents some interesting information extracted from the DBLP relational database.

Table 3.2 Some Statistics from DBLP database[2]

Statistics	Query	Result
Number of authors	Select count(*) from authors;	854332
Number of publications	Select count(*) from pubs;	2313000
Average number of authors in each publication	NA	5.4
Maximum number of publications have been done by one author	select distinct(author_id), count(*) as total from written_by group by author_id order by total desc;	579

[2] This statistics according to the last update of DBLP dataset on 1st October, 2010

Year has the most number of publications	select distinct(year), count(*) as total from pubs group by year order by total desc;	2008[3]

Figure 3.3 illustrates the dataset has evolved over the years in term of number of publications on the fields related to computer science research areas. Notice that since 1980 the number of publications has increased exponentially.

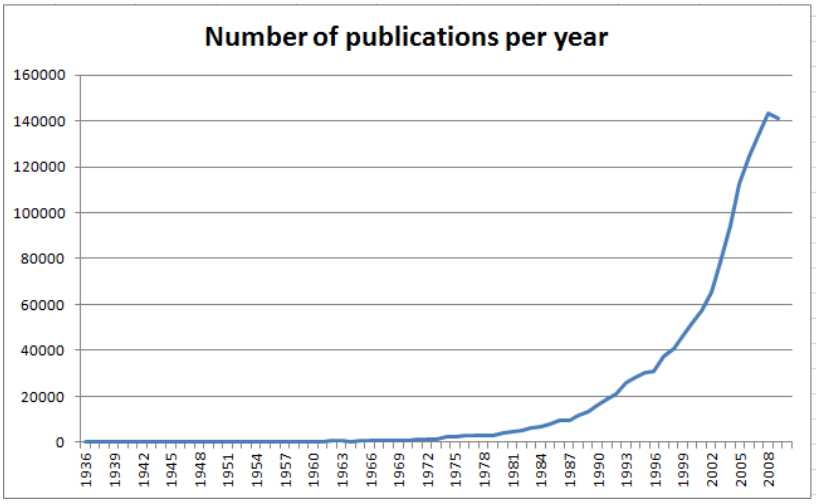


Figure 3.3 Number of publications which have been published since 1936

3.3 Enhancing DBLP Database with Affiliation Records

DBLP dataset can be enhanced by adding the affiliation information for each author to its records. Unfortunately this information is not available in the current DBLP dataset; therefore we have implemented an internet searching module using Web Information Retrieval [24] to retrieve this

[3] This result doesn't cover all publications, because some DBLP dataset records have no year element

information from the Web by using searching engines available on the Web such as Google[4] and Bing[5].

Google has developed its own searching API so-call SOAP Search API[6] for researchers who are interested in Google search but not for commercial use. SOAP Search API allows developers to execute up to 1,000 queries per day. According to this, we need around 850 days to get the affiliations of the DBLP authors, so that SOAP Search API is impractical solution for our case. Therefore, we developed our Search API which can execute more that this number of queries per day, despite the result will be not at the same level of accuracy and quality as Google's API, but this still more efficient way to get the affiliations in couple of days than waiting 850 days.

The steps are taken by our Search API model must be in the following order:

- Generate HTTP query contains author name in its script.
- Connect to one of the searching engines available on the Web.
- Send the search query, and then wait for result from the engine.
- Parse, extract and then filter the related links.
- Compare the result links to the institutions links from the database, and then update the affiliation of that author, if one of the results has been matched

Figure 3.4 shows a flowchart diagram of the Web information retrieval in our Search API, notice that the final decision depends on if one of the result links exists in the institutions links database which contains the links for 7843 universities in the world.

[4] Google is a free search engine owned by Google, for more information visit http://www.google.ca/intl/en/about/.
[5] Bing is a free search engine owned by Microsoft, for more information visit http://www.bing.com/.
[6] For more information about SOAP Search API visit http://code.google.com/apis/soapsearch

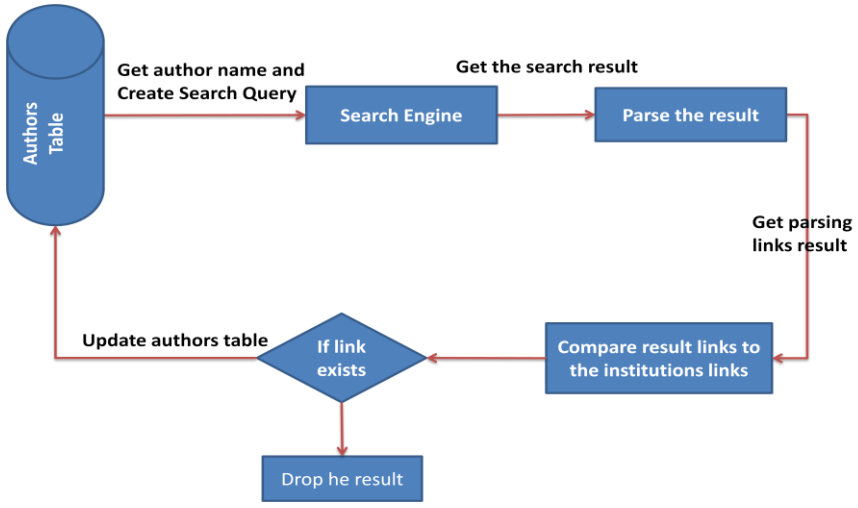


Figure 3.4 A flowchart diagram shows the process of Web information retrieval

With this new model each search request required 500ms to get the result back, therefore the time in milliseconds required to complete all search query requests is given by equation (3.1):

$$t = number\ of\ authors\ *\ request\ time \tag{3.1}$$

Based on that, we need around $850{,}000 * 500 * 10^3 = 118\ hrs$ to complete all search queries. Unfortunately, Google servers detected our machine behavior. So that, they blocked the software for couple of hours before we can start a new search request. Because of that we moved to Bing search engine, where we able to perform more than 20,000 requests per each run. As a final result, we have obtained about 53% of the affiliations of the authors in the DBLP dataset, and we have found that 3997 of 7843 universities are the current affiliation of the DBLP authors.

Affiliation records can be used to analyze the relationships among the authors according to their geographical location, or it can be used to study the relationships among the institutions all over the world.

3.4 Processing of the Tables

Social network graph structure consists of two major components, nodes and links. Each author in the DBLP social network represents a node, his coauthors represent a link, and the weight for each link represents the similarities or number of shared publications between any pair of authors, this information used to know how close these authors are to each other.

In order to build the co-authorship social network data structure which represents the node-link graph of the co-authorship social network, therefore we need to extract these coauthors relationships between the DBLP authors to build our co-authorship social network.

Written-by table in DBLP database schema has the relations between all DBLP authors and their publications which can be used to extract the relationships or number of shared publications between any pair of authors. Table 3.3 presents an example of a list of publication-authors, where we will explain the process of extracting coauthors relationships and to build the node-link graph.

Table 3.3 Publication-Authors list

Publication	Authors
P1	A1, A2, A3
P2	A1,A3,A4
P3	A1,A2,A3,A5
P4	A4,A6
P5	A1,A3,A5,A6
P6	A2,A4

Figure 3.5 shows a co-authorship social network for author A1 and his coauthors, each link connects pair of authors and it represents a publication they have coauthored, and the weight

indicates number of publications have been coauthored, this value represents the strength between social entities, therefore A1 and A3 are more closer to each author than A1 and A6 because they have coauthored more publications.

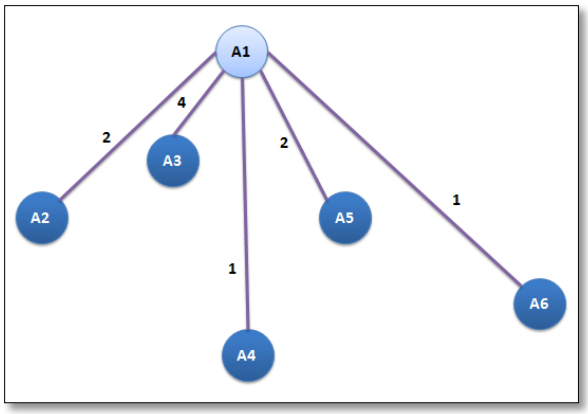


Figure 3.5 Node-Edge graph depict a co-authorship network

A mathematic way to represent the relationships in a social network graph is by using adjacency matrix which known as proximity matrix. The size of adjacency matrix for any graph with *n* nodes is *n* x *n*. Table 3.4 represents the generated adjacency matrix for our example. Notice that this adjacency matrix is sympatric because it represents an undirected graph.

Table 3.4 Adjacency Matrix for co-authorship network

	A1	**A2**	**A3**	**A4**	**A5**	**A6**
A1	-	2	4	1	2	1
A2	2	-	2	1	1	0
A3	4	2	-	1	2	1
A4	1	1	1	-	0	1
A5	2	1	2	0	-	1
A6	1	0	1	1	1	-

The link weight for any two authors those don't have a direct relationship is considered to be zero. A shortest path algorithm can help to calculate the weight between any two pair of authors. Due to that our co-authorship social network graph consists of more than 850,000 nodes, it is computationally infeasible - because the limit of our resources - to calculate the shortest path from each node to all nodes in the graph. In addition, building the adjacency matrix for this huge graph with more than 850,000 nodes is costly in term of the required memory space. The minimum size required to store an adjacency matrix with $n = 850{,}000$ is equal to 850,000 * 850,000 * 1 Byte per node/2 (Symmetric) = 361.25 GB.

In order to reduce the computations complexity and the space required of the co-authorship adjacency matrix, we have to take the following steps:

- Replace adjacency matrix by linked list (Edge Adjacency List) and instead of calculating the shortest path between all nodes, we calculate only the weight between any two direct connected nodes.
- The weight of indirect or not connected nodes is equal to zero.

If we take this rules, the size of the new structure will be 850,000 *(Nodes) * 6.54 (Average coauthors per any author) * 1 Byte per node = 5.56 MB.*

The format of the edge adjacency list is shown below:

Author ID, # of publications – coauthor$_0$ ID, # of publications – coauthor$_1$ ID, # of publications – coauthor$_2$ ID, # of publications - …

And the final representation of edge adjacency list for our example is:

A1, 4 – A2, 2 – A3, 4 – A4, 1 – A5, 2 – A6, 1
A2, 3 – A1, 2 – A3, 2 – A4, 1 – A5, 1
A3, 4 – A1, 4 – A2, 2 – A4, 1 – A5, 2 – A6, 1
A4, 3 – A1, 1 – A2, 1 – A3, 1 – A6, 1
A5, 2 – A1, 2 – A2, 1 – A3, 2 – A6, 1
A6, 2 – A1, 1 – A3, 1 – A4, 1 – A5, 1

Chapter 4

Multidimensional Scaling

4.1 Multidimensional Scaling

Multidimensional Scaling or MDS is a data analysis technique used in order to extract a set of independent variables from proximity data or matrices, and used to display or visualize distance-like information as geometrical points in 2D dimensions or transform the data to higher dimensions. MDS is considered one of the mathematics method used to visualize entities within social networks such as our co-authorship social network, it is applied on a set of data to approximate the distance between a pair of objects, these data called similarities, dissimilarities, distances or proximities. MDS algorithms have different types [8]. These types can be classified according to the input data matrix as following:

- **Classical MDS (CMDS):** in this type of MDS, the input data is one matrix represents the similarities or dissimilarities between objects, Classical MDS applies Euclidean Distance to find the similarities or the distance d_{ij} between an object i and j in the symmetric matrix X_{nxn} , which can be defined as in equation (4.1):

$$d_{ij} = \sqrt{\sum (x_{ia} - x_{ja})^2} \tag{4.1}$$

  Where x_i is ith row of X, x_j is jth column of X, and a refers to the dimension.

  A coordinate matrix is an output of CMDS. This matrix has a new configuration of the objects, and the objective of applying CMDS is to minimize a loss function called strain.

- **Metric MDS:** the input for this type of MDS is a distance matrix D_{nxn} is computed from Euclidean geometry, and the output is a new configuration of the objects in *p*-

dimensional space, where the elements along p dimensions are as close as possible to the elements in matrix D_{nxn}.

- **Non-metric MDS (NMDS):** in this type of MDS, we have two input matrices, one represents the proximities or dissimilarities between objects in a matrix, and the other represents the Euclidean distances between them. The purpose of NMDS is to find a configuration that minimizes the squared differences between the proximities and the distances between the geographic points so-called stress. See equation (4.2)

$$stress = \sqrt{\frac{\sum(f(p) - d)^2}{\sum d^2}} \tag{4.2}$$

Where $f(p)$ is scaled proximities which is known as *disparities* $\widehat{d}$, and d is the point distances.

4.2 Graph Visualization in 2D using MDS

MDS has become a desirable technique in social networks data analysis and visualization. This section explains the step to visualize the graph ***G*** of the co-authorship social network in DBLP dataset in 2D. Where the authors are represented as a vertices ***V*** have geometric locations (points) on ***X*** and ***Y*** axis, and the Euclidean distance between the vertices on the graph represent how close the authors are to each other.

The similarity between any two nodes (u, v) in DBLP dataset is measured by finding the number of publications $\#pub(u, v)$ they have coauthored. The proximity measure between any two nodes (u, v) in the graph is given by equation (4.3) :

$$p(u, v) = \alpha(w(u, v))^{-\beta} \tag{4.3}$$

Where:

$w(u, v)$ represents the weight of the link that connects two any nodes in the graph. α and β are two parameters used to modify the values of the proximities. Initially α=1 and β=1.

To find the best configuration of the nodes in the graph, stress $\boldsymbol{\sigma}$ between the nodes must be calculated and minimized. In co-authorship social network we defined the stress to be the squared differences between the proximities $p(u, v)$ and the Euclidean distance between vertices $\boldsymbol{V}$ in the graph. The Euclidean distance formula is given be the equation (4.4).

$$d_{u,v}(X) = \sqrt{(x_{i1-}\, x_{j1})^2 - (x_{i2} - \, x_{j2})^2} \tag{4.4}$$

Where $= \{(x_{1,}x_2),\ x_{1,}x_2\ \in X, 0 \leq X < 1000\}$, i is the index for node u, and j is the index for node v.

The total stress $\boldsymbol{\sigma}$ is given by the summation of individual stress for each node in the graph. See equation (4.5).

$$\sigma(X) = \sum_{i<j, w_{ij} \neq 0} w_{ij}(p(u, v) - \, d_{u,v}(X))^2 \tag{4.5}$$

A good configuration of the nodes on the graph $\boldsymbol{G}$ has minimum stress which can be calculated by using one of the optimization methods such as Steepest Descent also known as Gradient Descent method. This optimization algorithm helps us to reach our goal which is minimizing the overall stress of the vertices on the graph and forcing each node to move in the direction near the most similar nodes, this operation known as negative gradient.

Gradient Descent is considered as a First-order optimization method, and it is calculated iteratively to improve the configuration and to have a better layout of the nodes in the graph. Therefore, to find the minimum value of the stress function $\sigma(X)$, the partial derivative of it must be taken. See equation (4.6).

$$\nabla\sigma(X) = \frac{\partial\vec{\sigma}}{\partial x_{il}} \tag{4.6}$$

Where $l = 1,2\ and\ i = 1,2,3 \ldots n$.

By taking the partial derivative for the stress function will give to the following equations: 1) Equation (4.7.a) with respect to $x_{i1}$and 2) Equation (4.7.b) with respect to x_{i2}.

$$\frac{\partial\sigma}{\partial x_{i1}} = -2 \sum_{wij\neq 0} (p(u,v) - d_{ij}(x)) . \frac{1}{d_{ij}} . (x_{i1} - x_{j1}) \tag{4.7.a}$$

$$\frac{\partial\sigma}{\partial x_{i2}} = -2 \sum_{wij\neq 0} (p(u,v) - d_{ij}(x)) . \frac{1}{d_{ij}} . (x_{i2} - x_{j2}) \tag{4.7.b}$$

After computing the negative gradient from the optimization procedure above, the result will be applied on the previous configuration of the nodes to get a new configuration, thus the distances between nodes as close as to the their proximities.

$$x_{i1} \leftarrow x_{i1} - \gamma \frac{\partial\sigma}{\partial x_{i1}} \tag{4.8.a}$$

$$x_{i2} \leftarrow x_{i2} - \gamma \frac{\partial\sigma}{\partial x_{i2}} \tag{4.8.b}$$

Where $\boldsymbol{\gamma}$ is the step size parameter has the value between 0 and 1.

The steps must be taken to apply the nonmetric MDS algorithm are:

Nonmetric MDS Algorithm	
1:	Initialize the nodes on the graph with random configuration points X_{il}
2:	Compute the proximity measure $\boldsymbol{P}$ between the nodes. Equation (4.3)
3:	Compute the Euclidean distance between the nodes. Equation (4.4)
4:	Compute the stress $\boldsymbol{\sigma}$ for each node. Equation (4.5)
5:	Compute the partial stress using negative gradient, in order to minimize it between

the nodes by finding the new a configuration of the graph layout. Equation (4.6) and Equation (4.8.a)

6: Iterate to step 3 until the stress converged or become small enough.

4.2 Implementation and Preliminary Result of MDS

DBLP dataset has more than 850,000 authors and more than 2.7 million edges which mean that a graph visualization tool must be able to handle and draw all these nodes. Most of social network visualization tools have limitations on number of nodes that can be graphed because the processing time and the memory required. To deal with all these nodes and edges, we have to build our visualization tool that also support nonmetric MDS algorithm.

We chose Java[7] programming language as a platform to develop a graph visualization tool and to implement the nonmetric MDS algorithm, Java is one of the best object-oriented programming languages and has an intermediate performance. By using java, different methods and structures have been implemented in order to construct the MDS algorithm on DBLP co-authorship social network, and overcome the computational performance and graph layout problems that we had faced during this research.

4.2.1 Nonmetric MDS using Static Array structure (MDS-SA)

In this structure, the nodes of the graph $\boldsymbol{G}$ structure have been stored in a one-dimensional static array which provides a fast access to the data and requires less memory space than other structures. In addition we use the node ID as a reference to its location in the array, so that the access time to any node data is $\boldsymbol{O}(\mathbf{1})$.

[7] http://www.java.com/en/

The pseudo code for MDS-SA is shown below, and the overall performance of MDA-SA is given by equation (4.9):

$$T = l_1 + l_2 + 5.4kn \quad (4.9)$$

Where l_1 is constant time required to load node graph structure, l_2 is constant time required to load or initialize the configuration of the nodes, k is number of iterations required until the overall stress is optimized, n is number of nodes in the graph, and 5.4 is average number of co-authors per publication.

Running MDA-SA for k times required $\boldsymbol{O(kn)}$, iterations stop when the average of the stress for all nodes become small enough or when the stress converges after k iterations.

Nonmetric MDS Static Array (MDS-SA) pseudo code

Load Nodes Graph Structure $\boldsymbol{G}$ to static array of one-dimension.
Load or initialize uniform distributed random coordinates for nodes $\boldsymbol{X_{il}} = \{(x_{i1}, x_{i2}), 0 \leq x_{i1} < 1000, 0 \leq x_{i2} < 1000\}$ to static array of one-dimension.

```
while ( ||∂σ||_2 > ε)
    for each node u ∈ G
        get co − author nodes v ∈ G
        for each v ∈ G
            calculate disparities p(u, v)
            calculate Euclidean d_{u,v} (X)
            calculate partial stress σ_{u,v} (∂σ/∂x_{i1}, ∂σ/∂x_{i2}) and save the new configuration X_{il}
        end for
    end for
end while
```

Where $\boldsymbol{\varepsilon}$ is an optimization factor, indicates whether the MDS-SA iterations should stop or continue.

The object of applying MDS on co-authorship social network is to minimize the stress between all the nodes so that nodes with close similarities become as near as possible to each other, which will be reflected on the graph by showing all strongly related nodes on the same drawing area while the nodes with weak-relationships or with no-relationships will be drawn alone or away from other strongly related nodes. Figure 4.1 shows different configurations for the nodes in the graph[8] after running MDA-SA for $k = 1000, 2000, 3000$ and 4000 iterations. The stress value of any node is indicated by different color, black color indicates nodes with low stress while red color indicates nodes with high stress. MDS optimization aims to minimize the stress value so a good fitting solution can be achieved.

Form Figure 4.1, we can notice that nodes with strong-relationships move to be close to each other per iteration, and they have higher stress than nodes with weak-relationships, also during the MDS iterations process, we can observe the movement of the nodes with strong-relationship toward the center. The average stress of the graph started at $3.1008E^7$ and continued decreasing until stress became 0.62 after $k = 25{,}000$ iterations. See Figure 4.2 which shows the final configuration of the nodes on the graph with stress = 0.62.

Another approach to have better fitting solution is by using stress convergence therefore MDS iterations will stop once the *stress* function approaches a limit regardless to the slight changes in the stress which don't affect the overall nodes alignment on the graph. Equation (4.10) shows the formula for stress convergence.

$$||\partial\sigma||_2 = \sum_{i=1,2\ldots n} \sum_{l=1,2} \left(\frac{\partial\sigma}{\partial X_{il}}\right)^2 \qquad (4.10)$$

[8] The graph has *1000x1000* pixels resolution, which means that in average there is one node in each pixel, but during the MDS iterations, many nodes start mapping to the same pixel.

From the above equation $||\partial\sigma||_2$ should be less than or equal to ε in order to reach the convergence condition.

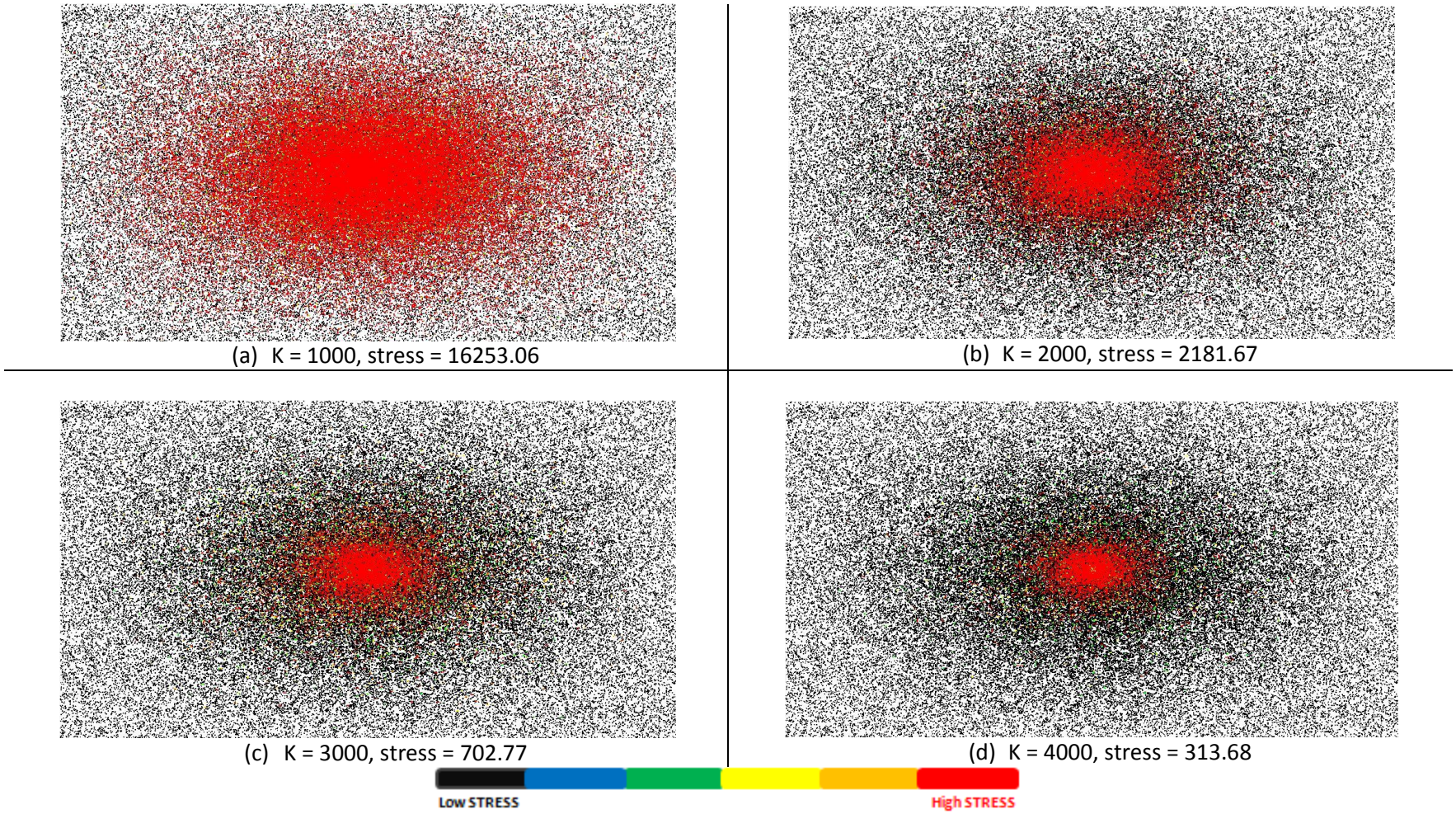


Figure 4.1 Snapshots showing DBLP nodes graph for different number of iterations

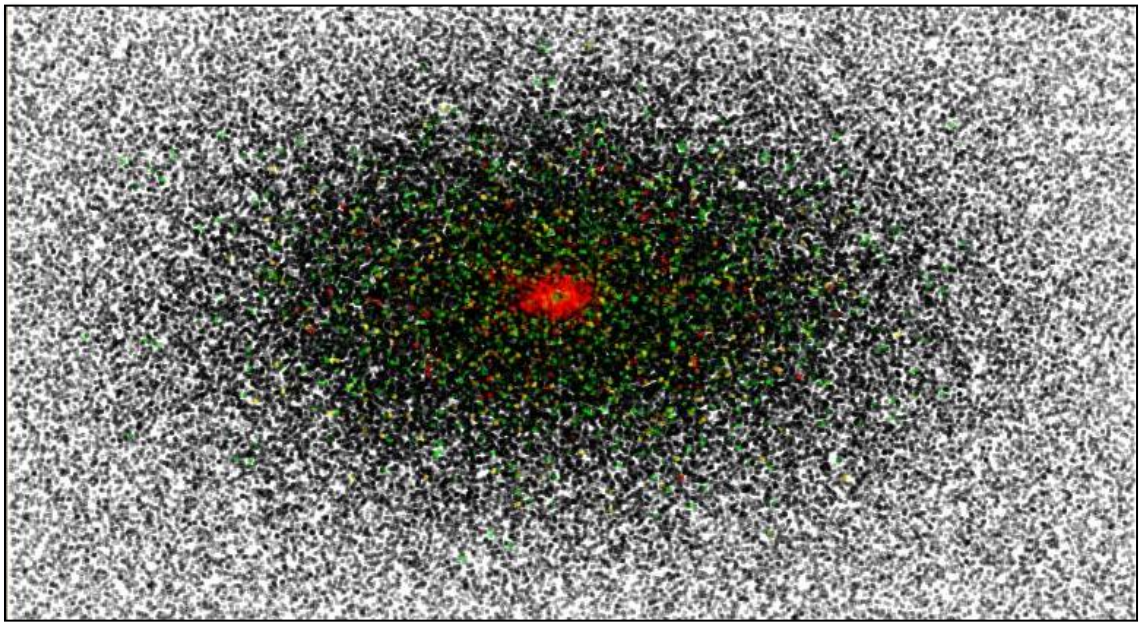


Figure 4.2 Co-authorship nodes graph with k = 25,000, stress = 0.62

Figure 4.3 shows the *stress* convergence for $k = 1000$ iterations. We take the logarithmic value of the stress to show a clear draw of the convergence. The simulation result shows that stress value decreases exponentially through the iterations.

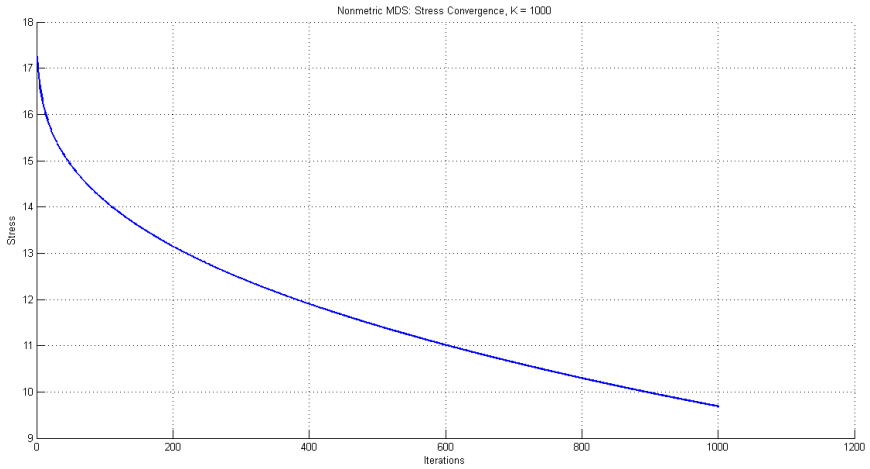


Figure 4.3 Stress convergence, k=1000

The step size $\boldsymbol{\gamma}$ in the direction of the gradient determents the speed of the stress convergence [25], the stress convergence process becomes slower for very small step size, and may require a

large number of iterations before reaching the local minimal. On the other hand, choosing a larger value for the step size can led the process to diver way from the local minimal. Figure 4.4 illustrates the speed of the stress convergence for different values of step size $\boldsymbol{\gamma}$. We chose $\gamma = 0.001$ in this research in order to maintain the stability and avoid overshooting the local minimal.

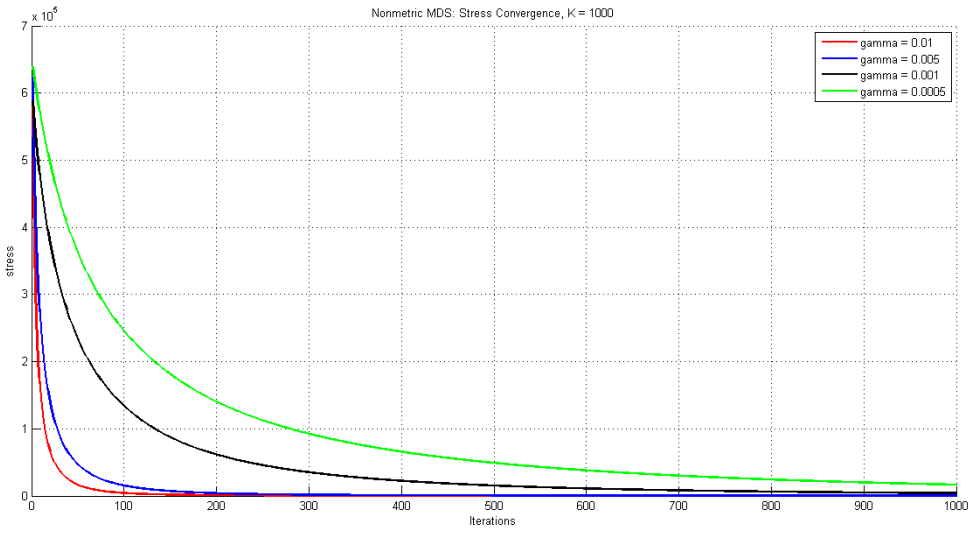


Figure 4.4 The stress convergence for different step size γ (gamma), $k = 1000$

The density of the nodes increases toward the center. Therefore, nodes that mapped near to the center have a higher stress and more strong-relationships with the closest nodes. See Figure 4.6 which shows the density of the nodes in the graph close to its center, many nodes mapped to the same area or to the same pixel, therefore it is impossible to understand the nature of the graph or the generated drawings without take a close look inside this dense graph. Figure 4.5 shows a zoomed in snapshot close to the graph center, you can see many cliques around the center; also some of them have a path toward the center.

Because the relationships between authors in a co-authorship social network depend on the publication they have coauthored and published, we predict finding variety of graph classes which are produced by nonmetric MDS algorithm; these sub-graphs in this network represent small communities which generally have common research topics or fields.

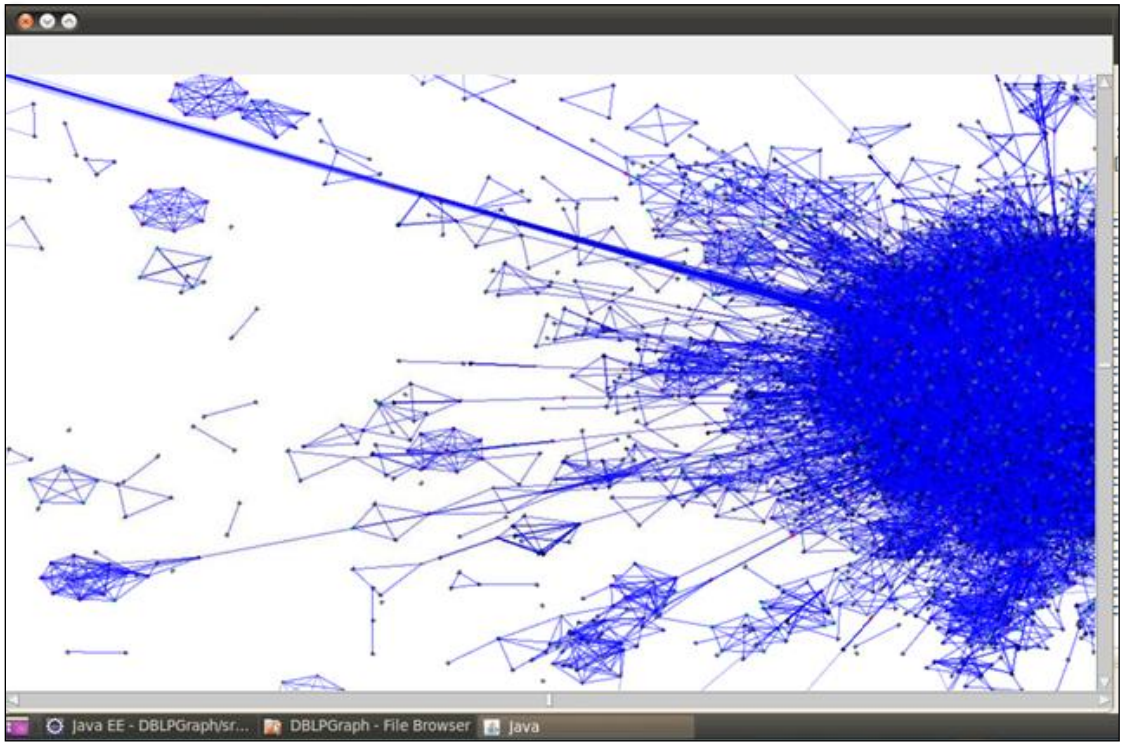
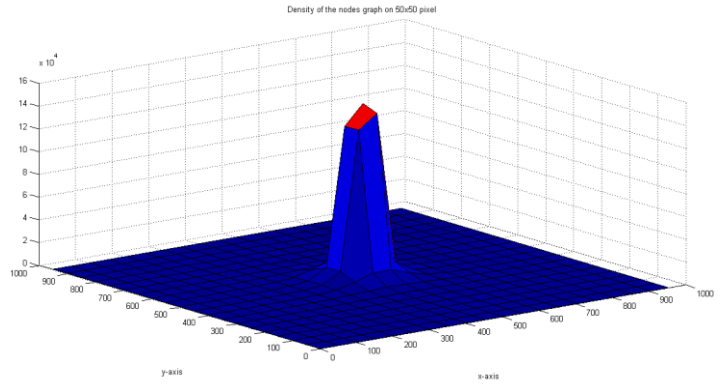
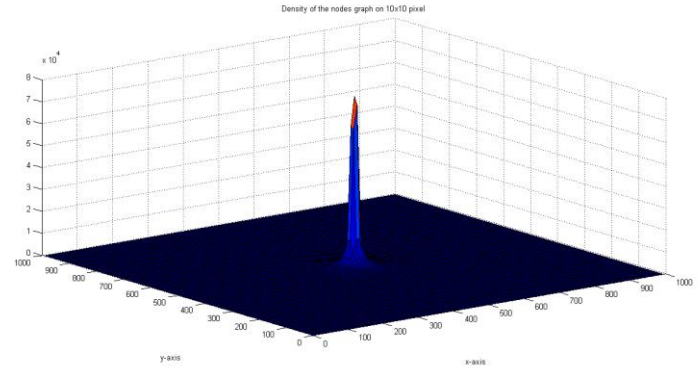


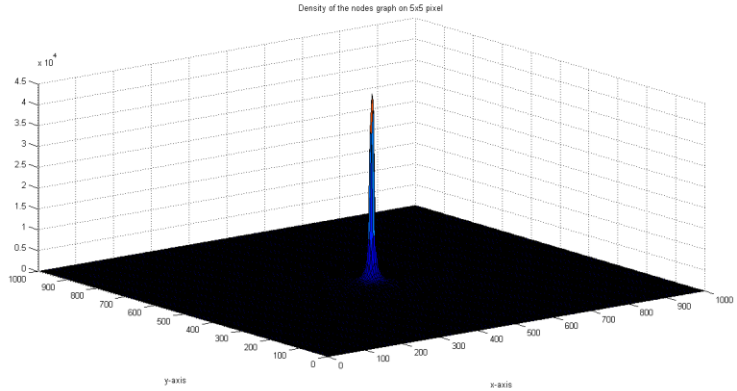
Figure 4.5 Close snapshot of the center of DBLP nodes graph



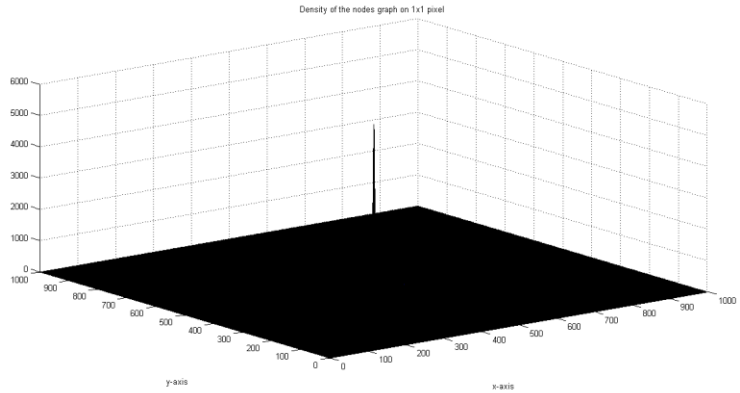
$(a)\ 50x50\ pixels, k = 1000, Density =\ 150272$



$(b)\ 10x10\ pixels, k = 1000, Density =\ 76437$



$(c)\ 5x5\ pixels, k = 1000, Density =\ 44804$



$(d)\ 1x1\ pixels, k = 1000, Density =\ 5030$

Figure 4.6 Nodes density on the graph for different area divisions

The process of stress optimization using steepest gradient descent causes the nodes with strong relationships and with high average co-authors to be attractive to each other, and be as close as possible. Unfortunately, optimization process of the stress may become stuck in a local minimum as a result of steepest descent convergence. In the following section, we will use simulated annealing [23] method to investigate if the nature of the co-authorship social network graph causes this attraction of the nodes at the center of the graph.

4.2.2 Nonmetric MDS using simulated annealing (MDS-SAN)

Steepest gradient descent in the long run converges to a local minimum [9], which may be not the best layout of the nodes. Simulate annealing is used to locate a solution for global optimization problems such we have with the graph stress, simulated annealing algorithm start with a random solution where it's iterations generate a new solution from the last one and replace it, if it better than the old one.

The idea behind using simulated annealing is to find a better layout of these dense nodes mapped at the center of the graph, this process done by randomly spreading these nodes away from the center each time the stress converges, then repeat this several time until we have better solution. For each convergence iteration nodes are spread away from the center by a factor, and then decrease this factor by half after each convergence. The MDS-SAN pseudo code is shown below.

Nonmetric MDS with simulated annealing (MDS-SAN) pseudo code

Load Nodes Graph Structure $\boldsymbol{G}$ to static array of one-dimension.

Load or initialize uniform distributed random coordinates for nodes $\boldsymbol{X_{il}} = \{(x_{i1}, x_{i2}), 0 \leq x_{i1} < 1000, 0 \leq x_{i2} < 1000\}$ to static array of one-dimension.

$while\ (I > 0)$

    $while\ (||\partial\sigma||_2 > \varepsilon)$

        $for\ each\ node\ u \in G$

$get\ co-author\ nodes\ v \in G$

$for\ each\ v \in G$

$calculate\ disparities\ p(u,v)$

$calculate\ Euclidean\ d_{u,v}\ (X)$

$calculate\ partial\ stress\ \sigma_{u,v}\left(\frac{\partial\sigma}{\partial x_{i1}}, \frac{\partial\sigma}{\partial x_{i2}}\right) and\ save\ the\ new\ configuration\ X_{il}$

$end\ for$

$end\ for$

$end\ while$

$Randomlly\ Spread\ all\ nodes\ u \in G\ which\ are\ have\ \boldsymbol{X_{ul}} = \{(x_{u1}, x_{u2}),\ 490 \leq\ x_{u1} <$ $510,\ 490 \leq xu2 < 510.$

$withen\ circle\ of\ radius\ r = \frac{r}{2}$

$decrease\ number\ of\ remaining\ iterations\ by\ 1$

$end\ while$

Where r indicates the circle radius of the spreading area, and I is the number of simulated annealing iterations. initially r is chosen to be equal to half of the graph size.

Figure 4.7 shows the stress convergence through four simulated annealing iterations, stress still converge close to the last values which means that we still have the same dense layout of the nodes at the graph center.

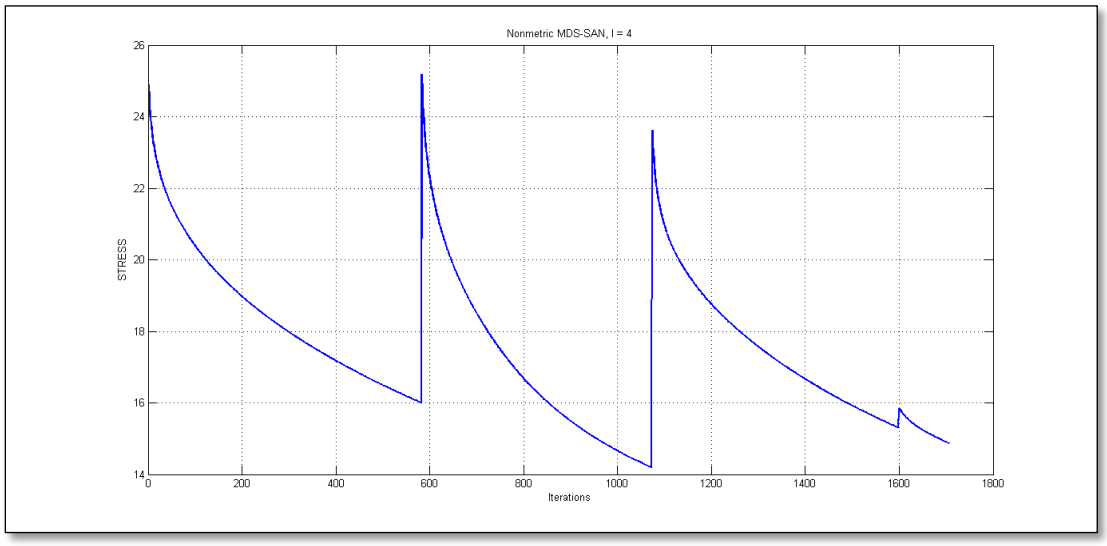


Figure 4.7 Nonmetric MDS-SAN with I = 4

Even by using simulated annealing to find out if the attraction of the nodes at the center is formed because of the nature of the social network relationships or because of the non-completed proximity matrix. Our proximity measures are calculated from one author to all his co-authors not to all authors in the network, because it is computationally infeasible to find all shortest paths between 85,000 nodes, therefore, we predict to have different graph layouts for the same network. Figure 4.8 shows how non-complete proximity measures of a social network can lead to have different node alignments on the same graph. This happens for the reason that we consider the weight between any two non-collaborated authors to be zero, so that it is possible to find many authors with zero weight close to each other in the graph.

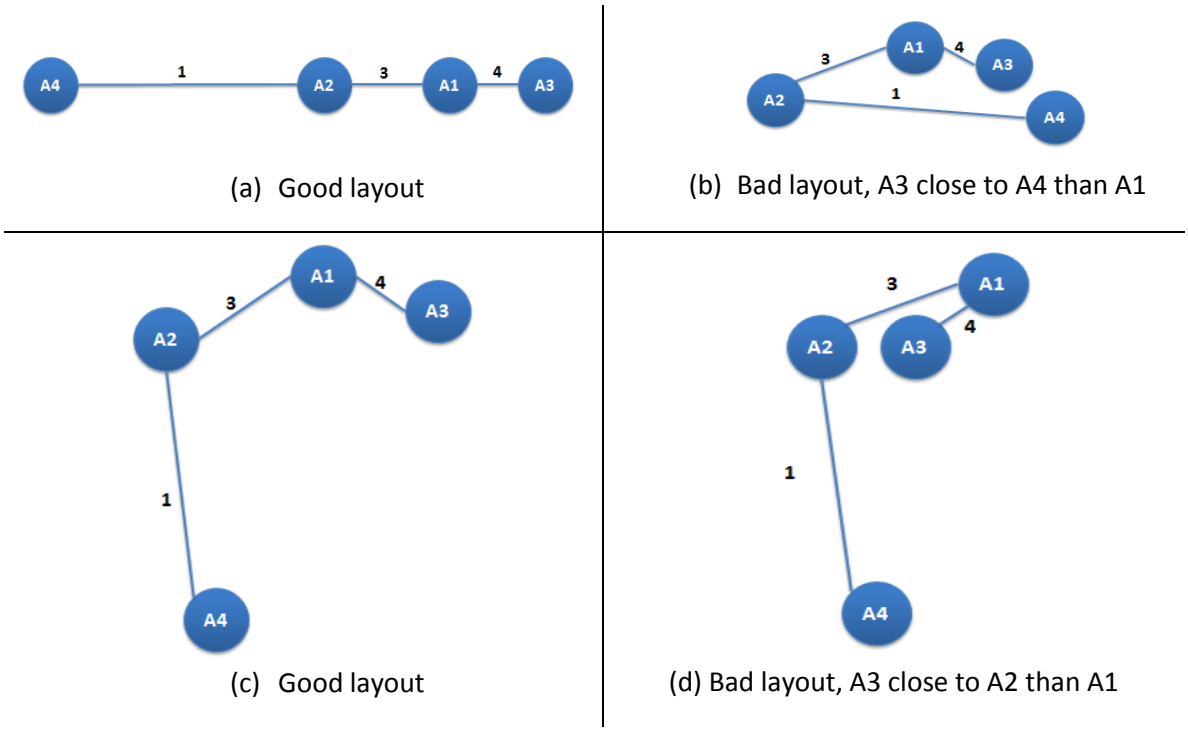


Figure 4.8 The same social network with different layout alignments.

To get rid from this problem, we propose a new solution in the following section where we add a repelling force between non- collaborated authors or these nodes without a direct link between them. As a result, nodes will be forced to move away from the non-direct connected nodes.

4.2.3 Nonmetric MDS with Repelling Force Method (MDS-R)

Forcing the nodes whose have no direct co-authorship to move away from each other on the graph, therefore a suitable layout can be obtained from the MDS algorithm. Repelling force is calculated using Euclidean distances between any two nodes have no direct link among them until reach stability.

The process of repelling is calculated by assuming there is a small weight $w_{u,v} < 1$ between nodes(u, v), where one is the minimum edge weight for any two connected nodes, according to

this assumption the dissimilarities $p_{u,v}(x)$ become larger than the other links which cause the repulsion. Figure 4.9 shows the repelling process of node A3 and A4 from their unconnected nodes, where $p_{A3,A2}(x) > p_{A3,A1}(x)$. For this case, after applying a repelling force, the new expected position for node A3 and A4 should be similar to graph (a) or (c) in Figure 4.8.

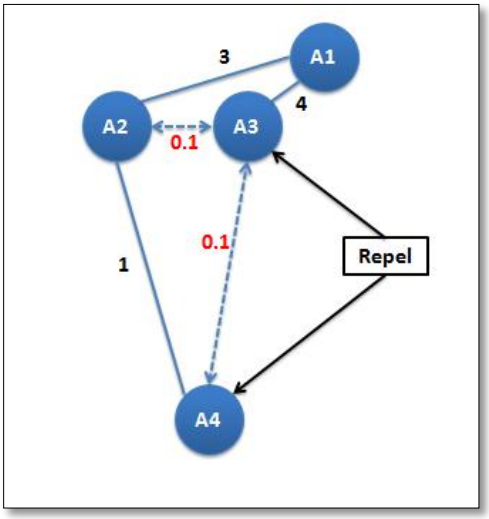


Figure 4.9 Repel node A3 and A4 away from other non-direct connected nodes

The process of nonmetric MDS with repelling force includes additional step, for each node $u \in G$ we have to calculate the proximity $p(u, v)$ of the direct connected nodes $v \in G$, otherwise if node v is not directly connected to node u, then calculate the proximity $p(u, v)$ with edge weight $w_{u,v} < 1$, i.e. $w_{u,v} = 0.1$, and then compute the Euclidean distances and the gradient on the stress. In other hand, the new updated version of nonmetric MDS must go through all nodes in graph G, which will significantly affect the overall performance of the algorithm. The pseudo code for the new version is shown below.

Nonmetric MDS with repelling force (MDS-R) pseudo code

Load Nodes Graph Structure $\boldsymbol{G}$ to static array of one-dimension.

Load or initialize uniform distributed random coordinates for nodes $\boldsymbol{X_{il}} = \{(x_{i1}, x_{i2}), 0 \leq x_{i1} < 1000, 0 \leq x_{i2} < 1000\}$ to static array of one-dimension.

$while\ (\ ||\partial\sigma||_2 > \varepsilon)$

$for\ each\ node\ u \in G$

$for\ each\ v \in G$

$if\ v \in coauthors(u)$

$calculate\ disparities\ p(u,v)$

$else$

$calculate\ disparities\ p(u,v), w_{u,v} < 1$

$calculate\ Euclidean\ d_{u,v}\ (X)$

$calculate\ partial\ stress\ \sigma_{u,v}\left(\frac{\partial\sigma}{\partial x_{i1}}, \frac{\partial\sigma}{\partial x_{i2}}\right)\ and\ save\ the\ new\ configuration\ X_{il}$

$end\ for$

$end\ for$

$end\ while$

For each nodes on the graph G, we have to calculate the repelling force for all indirect connected nodes, which has $O(n - 5.4)$ where 5.4 is average number of coauthors or average number of direct connected nodes on the graph G, as a result the overall performance of nonmetric MDS with repelling force will be $O(kn^2)$ where k is number of iterations. In this case implementing this algorithm will take a long time due to the limited resource we have.

To solve this issue, we propose another close solution where we divide the original graph G into cells C or into sub-graphs with equal sized area A, based on that each node u has to belong to one cell $c_u \in C$, and it can leave its original cell to another one or move to its neighbor cells.

Figure 4.10 illustrates how we divide the graph G into equal size cells where each cell has its own nodes, so that the repelling force can be only applied on all nodes belong to the same cell.

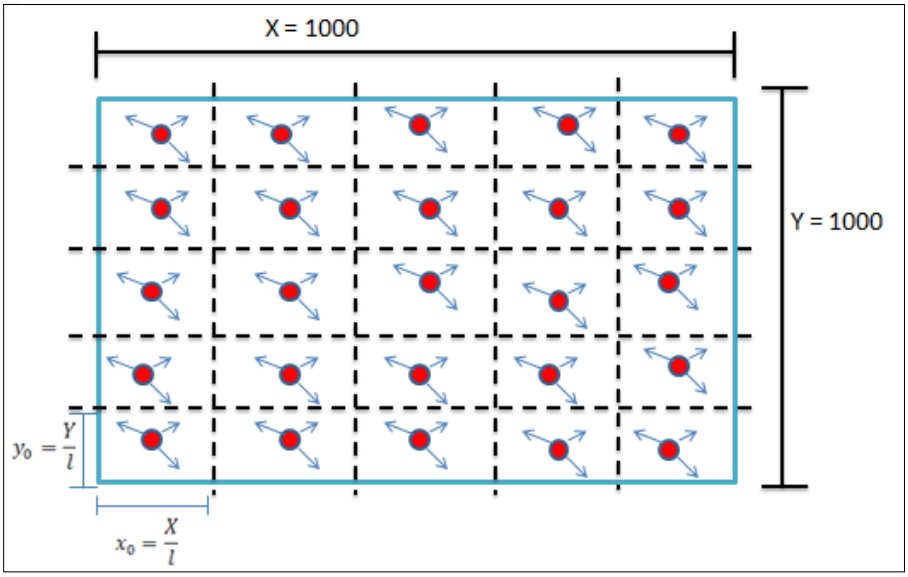


Figure 4.10 Dividing the bounding graph area into equal size cells

The repelling force will be calculated just for these nodes belong to the same cell not to all nodes on the graph which may produce the better graph layout. The overall performance depends on the average number of nodes per cell $\overline{n_c} = \frac{\#\ of\ nodes}{\#\ of\ cells}$, therefore the overall performance of this new approach is proportional to $\overline{n_c}$. Accordingly, we need $O(k\,\overline{n_c}$ n) to implement this approach.

Figure 4.11 shows the new nodes layout on the graph G for $k = 1000$ iterations after dividing the boundary into 500x500 cells. As a result, all strongly related nodes gather in the center of each cell which represents a small sub-graph of the whole graph G, these small sub-graphs represents small communities in co-authorship social network.

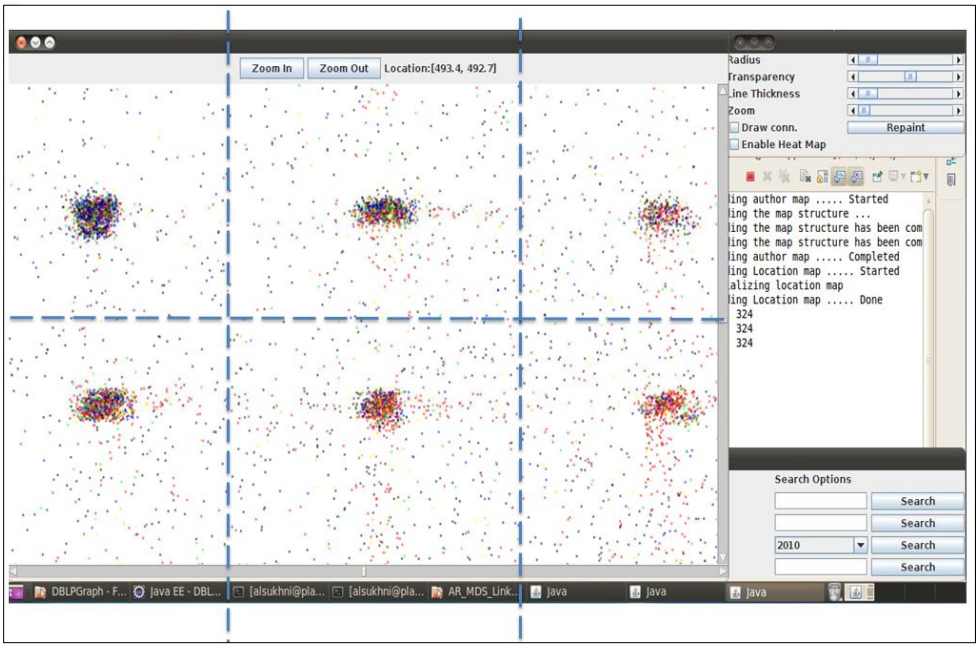


Figure 4.11 Nodes graph layout with 500x500 cells of 2x2 pixel cell size

The convergence speed of the stress depends on the number of cells in the graph G. Convergence speed is proportional to the number of the sub-graphs, and this can be seen in Figure 4.12 which shows that graph with 800 * 800 cells converges slightly faster that other graph divisions.

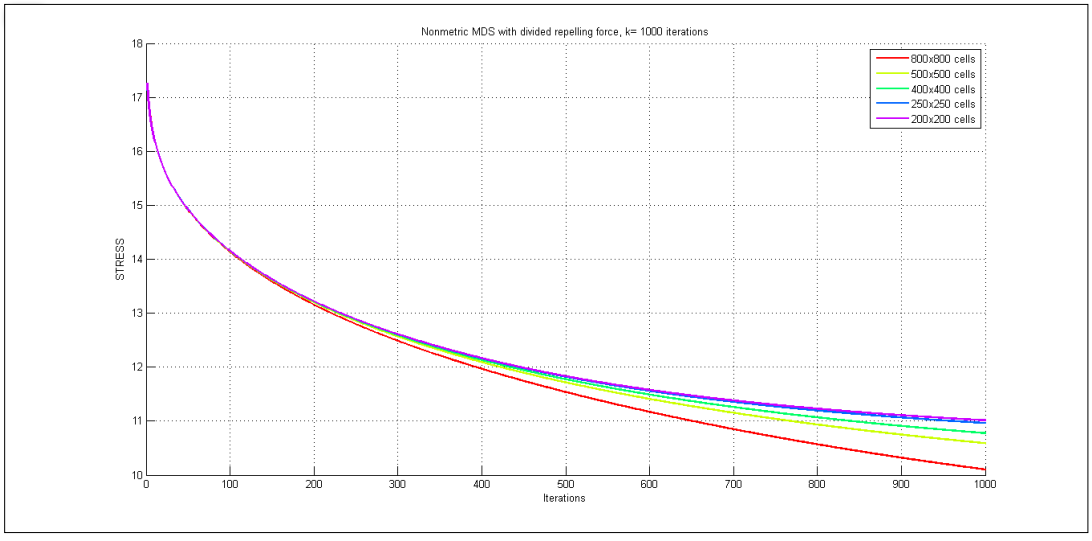


Figure 4.12 Stress convergence for different number of cells on graph G

This relation in convergence happens because when we have small size cells, nodes have more flexibility to move out from one cell to another cell. This means that nodes can faster form their own community. If we take a look to the time is required by nonmetric MDS iterations for different number of cells, we can notice that execution time increases during the iterations for all divisions, but nonmetric MDS execution time is inversely proportional to the number of cells, because many nodes start mapping to the same cell during the iterations which requires more repelling force calculations on these nodes. See Figure 4.13.

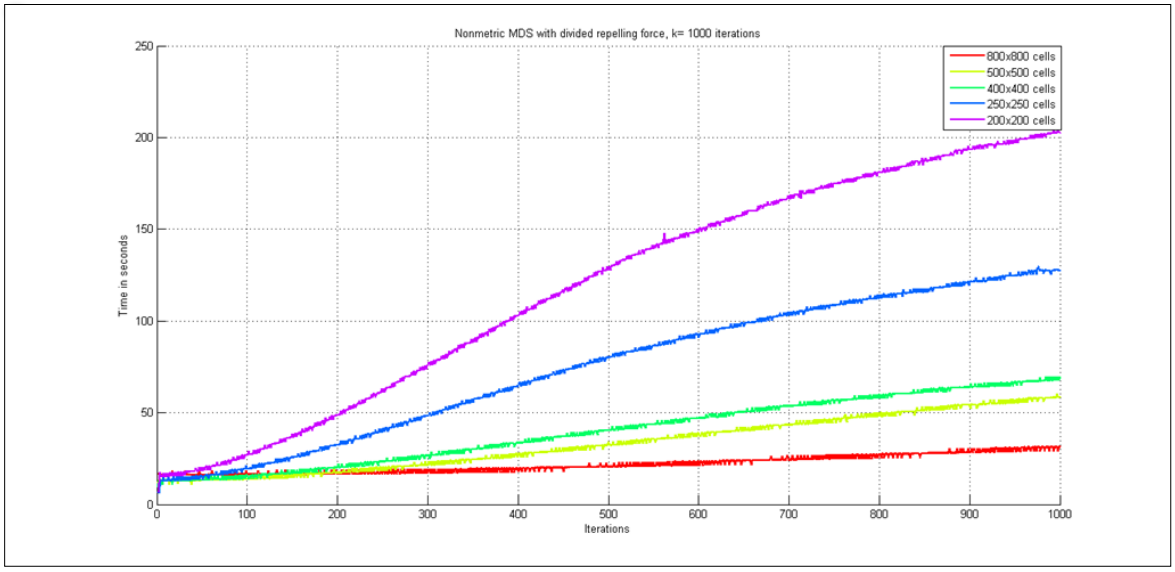


Figure 4.13 Execution time per iteration for different number of cells.

4.3 Results and Performance Analysis of Nonmetric MDS

We have implemented three different data structures to store the DBLP co-authorship network in order to find the best structure which improves the efficiency of nonmetric MDS algorithm. In section 4.2.1 we talked about how to implement nonmetric MDS algorithm using static arrays as a data structure for BDLP co-authorship graph network, and how static arrays are faster in term of access time and memory space required than other structures that if we use the node ID as a

reference index. We have implemented two another structures: a structure by using a hash map and another by using a custom linked list.

Hash map or hash table is a type of data structure where we use a hash function to map the keys to their values in the table. Hashing operation has $O(c)$ addressing time where c is a constant number, which means that hash map needs a constant time to look up a value from the hash table. Hash map is more efficient than other table lookup structure especially for large number of entries, but it requires more memory than static arrays.

In our implementation, we used the node ID as a key to store the nodes information, which helps us in applying the nonmetric MDS on a variety number of random nodes regardless the relation between them, while is in static array we have to reserve the nodes ID as a reference index. Therefore, it becomes more complex to choose any random number of nodes to apply MDS on them without have a conflict with their coauthors.

Table 4.1 and Figure 4.14 show the time required to minimize the stress value for variant number of randomly selected nodes, as a result we can notice that stress optimization runtime increases linearly with the number of nodes.

Table 4.1 Stress optimization required time by using hash map

# of nodes	Optimization Time	stress
4962	561.6	1.46E-25
10196	1702.865	7.42E-05
15103	2695.027	7.42E-05
20317	3327.406	0.00424224
25601	3512.003	0.01005449
30083	4149.718	0.00487824

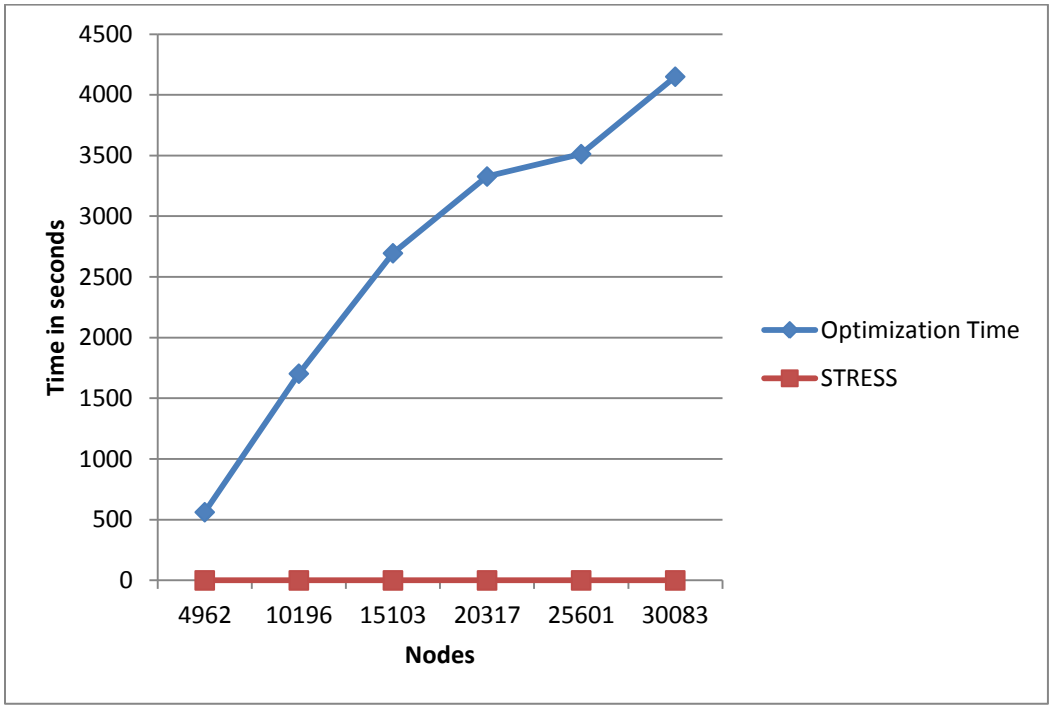


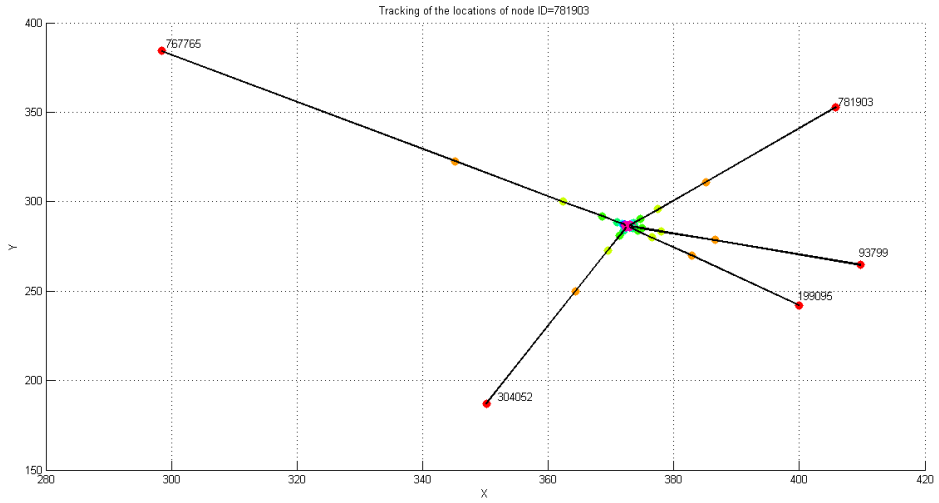
Figure 4.14 MDS-Hashmap: Optimization time for different number of nodes

Another way to evaluate the stress convergence is by tracking the location of one author and his coauthors on the graph, and observing how the distances between them become closer during the MDS iterations. Table 4.2 shows the initial random locations of five authors they had shared the same publication with edge weight equal to one among all of them. We will track node (id=781903) and then see how the MDS iterations minimize the stress among them by decreasing the Euclidean distances between them.

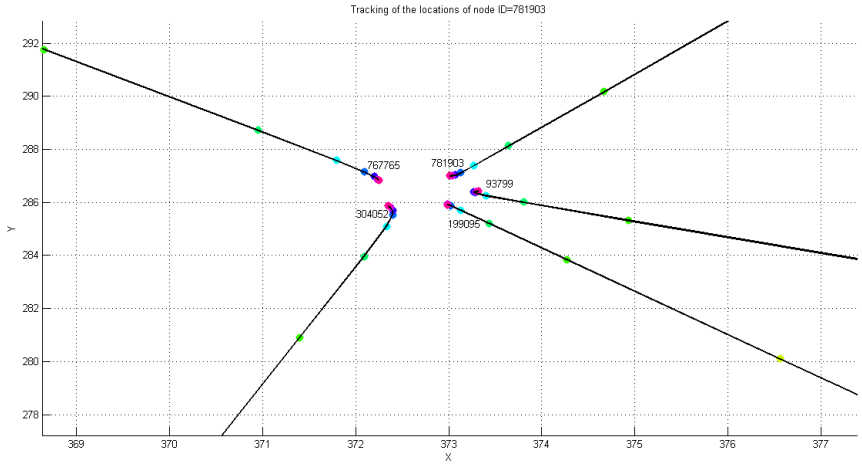
Table 4.2 Initial nodes configuration

ID	Initial X	Initial Y
781903	405.8339	352.8782
93799	409.7039	264.9384
199095	400.0050	242.1581
304052	350.1716	187.1702
767765	298.3989	384.2664

Figure 4.15 shows the tracking path for each node from its initial to the final location, nodes start with red color and the color of the nodes keeps changing during the iterations and end up with purple. From changes in the distances between the nodes in (a), we can notice that stress converges faster at the beginning of the iterations then becomes slower at the end. (b) Shows the final nodes alignment in the graph after $k = 1000$ iterations.

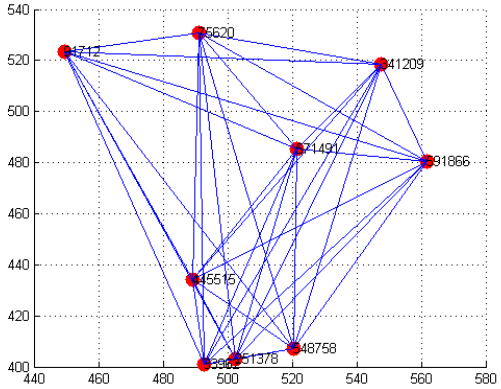


(a) All graph boundaries shown, zoom out

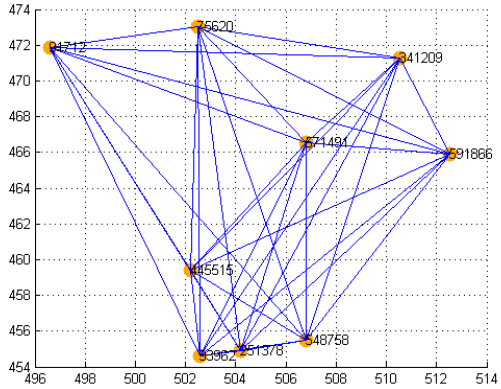


(b) Graph center, zoomed in

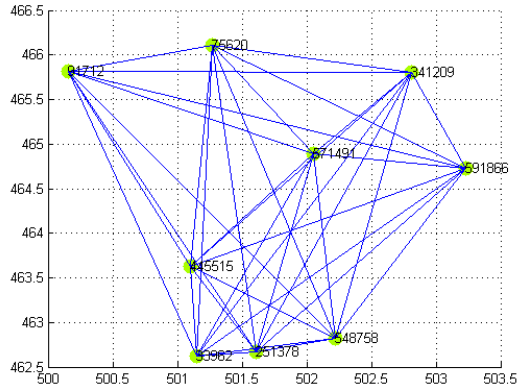
Figure 4.15 Location tracking of node id=781903 and its direct connected nodes on graph *G*.



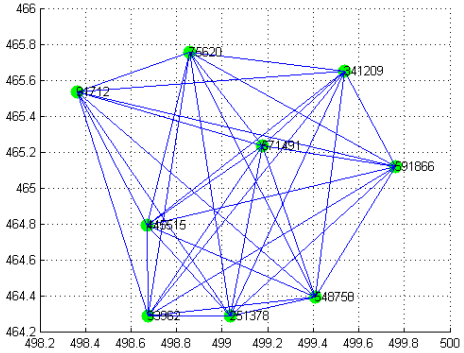
(a) Average stress = 7821.02, k= 100



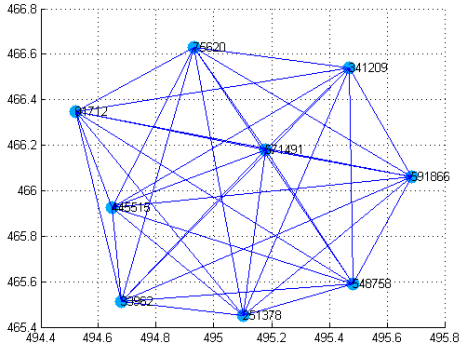
(b) Average stress = 171.77, k= 200



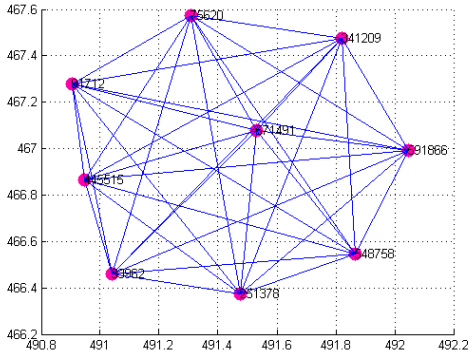
(c) *Average stress = 28.05, k= 300*



(d) *Average stress = 17.38, k= 400*



(e) *Average stress = 7.83, k= 600*



(f) *Average stress = 2.44, k= 900*

Figure 4.16 Evolution of a complete graph during MDS iterations

Figure 4.16 shows another example of formation stages of a complete graph in DBLP co-authorship social network during the MDS iterations, each graph represents different nodes layout which are generated after each k iterations, notice that the final clear and stable shape of a complete graph is generated after $k = 900$ iteratios.

In order to enhance the performance of nonmetric MDS and the calculation of repelling forces among the nodes on graph G, we have implemented a new data structure for this graph where we use a linked list to store the graph structure. Nodes in linked list are connected by using links, so each node has a reference to the next connected node.

In our implementation we divided the graph boundaries into cells in order to facilitate the process of repelling force calculation, where each cell has a separate linked list contains all nodes belong to this it, also each cell can have its own limited number of nodes per linked list, and nodes can move from one cell to another during the iterations. During the process, we go in sequence through the cells in the linked list instead of going sequentially through all the nodes in the iterations like in the previous implementations. Base on this, we save the lookup time required to find the node cell. Figure 4.17 shows a comparison between linked list and hash map structures for a different number of nodes, for small number of nodes the hash map is more efficient than our linked list due to number of linked lists created after dividing the graph into cells. Therefore, iterations will take more time in checking each cell, but for large number of nodes, linked list is more efficient than hash map.

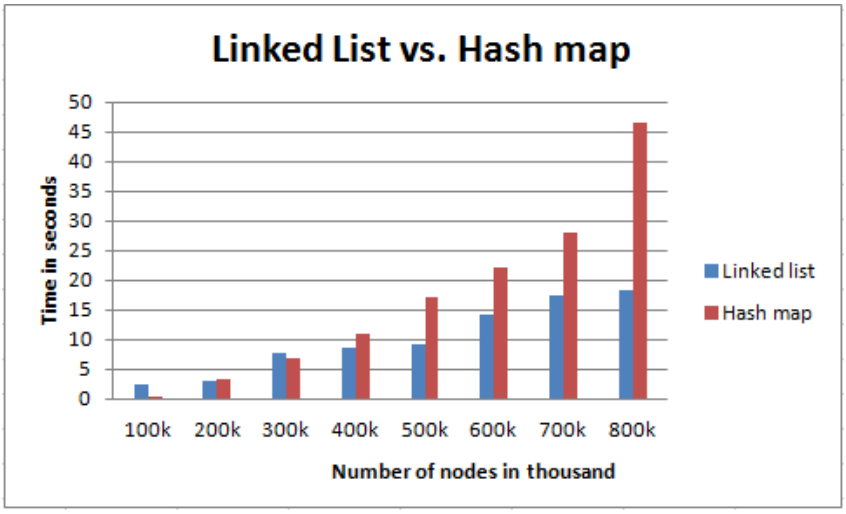


Figure 4.17 Time comparision between linked list and hash map

Figure 4.18 illustrates the runtime required per iteration through using three different types of structures. The figure shows that static array has the best in lookup time – in case we use the node ID as a reference to its index – while the hash map has the worst lookup time, and the linked list is in between. Therefore, static arrays are used when you have to apply MDS on all nodes, while linked list is much efficient for selected nodes or for a small portion of the graph.

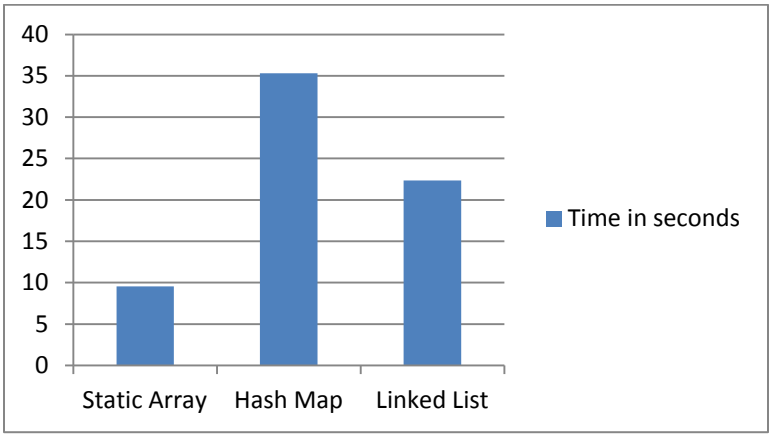


Figure 4.18 Runtime required per iteration for different data structures

Chapter 5

Multithreaded Nonmetric MDS

MDS optimization aims to minimize the stress value of the nodes on the graph in order to find the best layout for these multidimensional data. The cost of stress minimization using steepest gradient descent is $O(kn)$ which is considered high cost according to DBLP co-authorship social network size. The time required to complete MDS iteration for one time is shown in Figure 4.18. Stress optimization process goes through sequential stages where we have to find the stress value for each node and then minimized it by finding a new configuration for it and so on for the rest of the remaining nodes. We have implemented a multithreaded version of nonmetric MDS algorithm in order to speed up the optimization process and improve its efficiency.

Multithreading [26] allows more than one thread to work independently in the same single process, so that each thread can concurrently perform its task with other threads, but in some cases, these threads may share the same data and they may require to modify these data, so that synchronize the access to the data become critical issue if these threads have to modify it. Nodes configuration is considered the main shared data among all threads where each thread has to access and modify it, while the process of calculating the node stress and the partial stress is entirely independent. Therefore, we should synchronize the modification step of the stress optimization which is shown in equation (4.8.a) and equation (4.8.b).

Our idea is to distribute all the nodes of the graph among n threads, and then synchronize any write operation to the graph structure. According to this assumption, each thread t is responsible to perform the gradient descent only on its partition and then update the new configuration for each node. Figure 5.1 illustrates the flowchart diagram for multithreaded nonmetric MDS

algorithm where all threads must collaborate to minimize the stress, also notice that all threads must wait to join each other after completing each iteration in order to keep the consistency of the nodes configuration, so that all threads work in the same iteration.

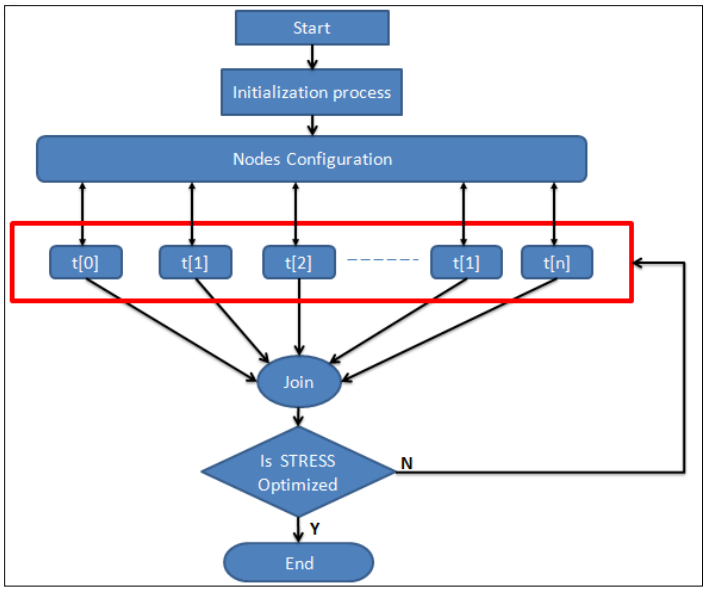


Figure 5.1 Multithreaded nonmetric MDS flowchart diagram

The race condition happens when unsynchronized threads try to modify data while another thread is working on it, also unsynchronized parallel threads which can exist in any iteration during the optimization process may cause data to be unstable and may slow down the stress convergence. A pseudo code for multithreaded nonmetric MDS algorithm is shown below.

Multithreaded Nonmetric MDS Pseudo Code

Load Nodes Graph Structure $\boldsymbol{G}$ to static array of one-dimension.

Load or initialize uniform distributed random coordinates for nodes $\boldsymbol{X_{il}} = \{(x_{i1}, x_{i2}), 0 \leq x_{i1} < 1000, 0 \leq x_{i2} < 1000\}$ to static array of one-dimension.

Create threads $\boldsymbol{t}[\boldsymbol{n}] \in \{\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3} \ldots \boldsymbol{t_n}\}$

$\boldsymbol{while}$ ($||\boldsymbol{\partial\sigma}||_{\boldsymbol{2}} > \varepsilon$)

$$For\ each\ thread\ t_k\ \in \{t_1, t_2, t_3 \dots\ t_{n1}\}$$

$$nonmetricMDS(X_{il}, t_k)$$

$$update\ nodes\ configuration\ X_{il}$$

$$end\ for$$

$$end\ while$$

Figure 5.2 shows the time required to run 100 iterations for a different number of threads, our multithreaded nonmetric MDS has been implemented under 2.66GHz cores Ubuntu machine. The result shows that the performance increases for two threads because the two cores are utilized, and then become stable until we reach 10 threads then performance started to decrease because of the overhead caused by these threads.

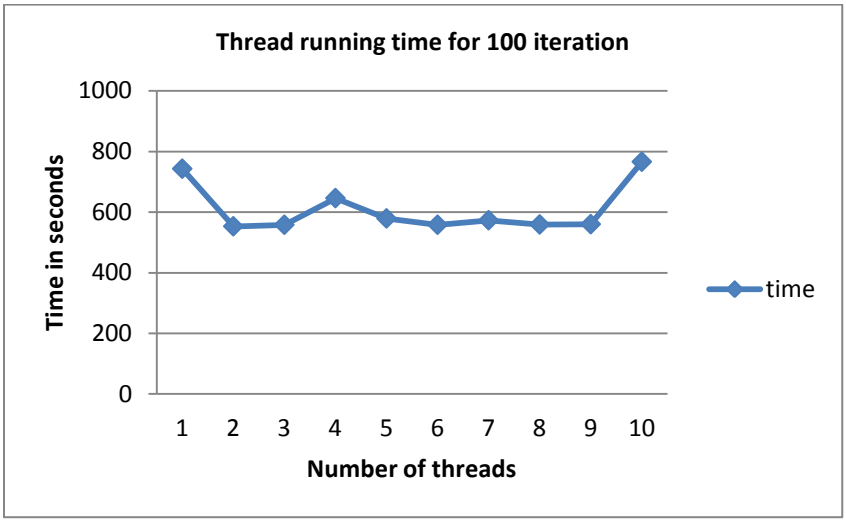


Figure 5.2 Nonmetric MDS performance for a different number of threads

Chapter 6

DBLP Graph VIS

Since we have more than 850,000 nodes and more than 2.7 million links on DBLP co-authorship social network graph, it seems difficult to any available graph visualization tool to handle all these nodes on one graph. Therefore we have designed and implemented our social network visualization model using Java API libraries such as Swing and AWT libraries, and we have provided the user with several important features that can help him to navigate and explore the DBLP social network graph such these features are: searching options, graph scaling, and drawing options.

In the following sections we are going to describe the input format of DBLP Graph VIS, explore all its features, and address some limitations.

6.1 DBLP Graph VIS Input Format

Our visualization tool is responsible to draw the final nodes configuration which came as result of implementing nonmetric MDS algorithm. The input file of the nodes configuration must be in the following format (*node ID, x coordinate, y coordinate*), also it is important to provide the software with the edge adjacency list which we have illustrated it in Chapter 3 section 3.4 in order to draw the edges among the nodes.

The node coordinates are preferred to be in double precision format, thus the nodes with high precision format will not be mapped to the same pixel on the graph when graph scaling technique.

6.2 Graphical User Interface (GUI) of DBLP Graph VIS

The graphical user interface (GUI) of DBLP Graph VIS has been designed to give the ability for the user to exploit the software features in easy and comfortable way. Figure 6.1 shows the GUI of our visualization tool which consists of three main frames each one has different functionalities.

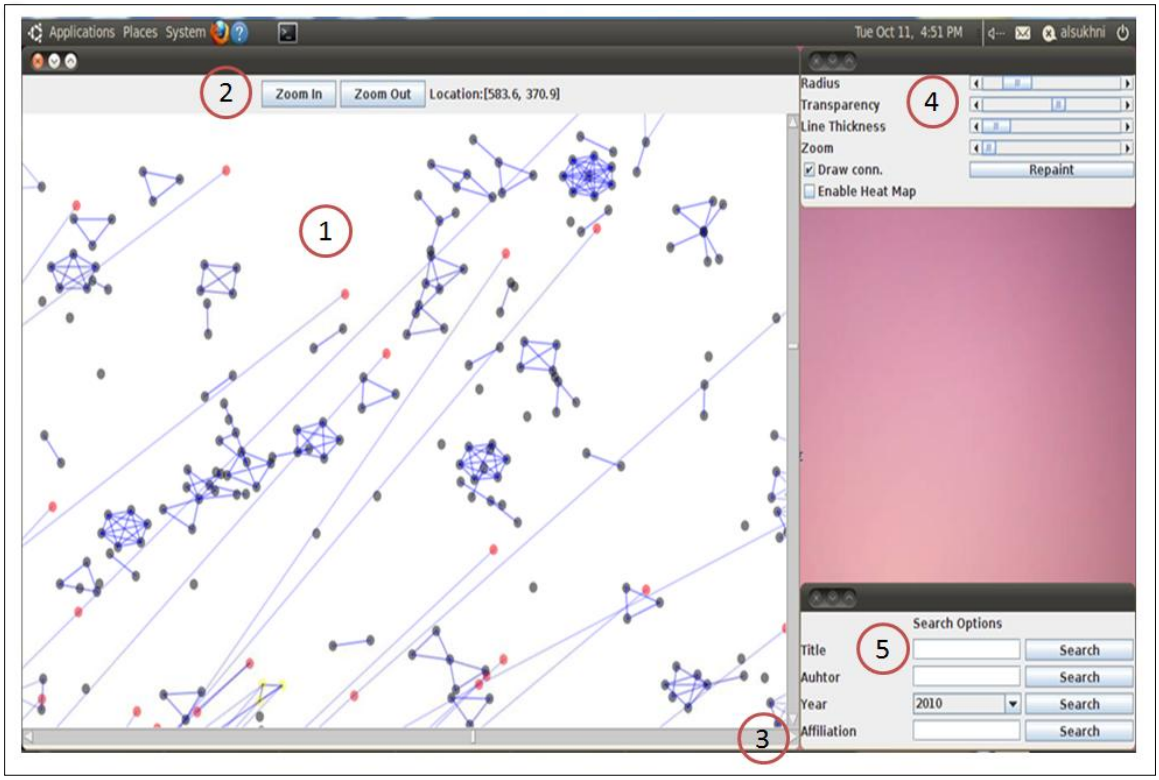


Figure 6.1 Graphical user interface (GUI) of DBLP VIS

The main components of the graphical user interface of the DBLP graph VIS are:

1. Drawing area: this area is designated to draw the graph nodes with the edges or links, each node represented as a single colored circle, the coloring code indicates the goodness of the node stress σ as show in Table 6.1.

Table 6.1 Coloring code for different values of stress

Color	Stress σ Perc. according to average value	Goodness of the stress
Black	$\sigma \leq 33.3\%$	Perfect
Blue	$33.3\% < \sigma \leq 66.6\%$	Excellent
Green	$66.6\% < \sigma \leq 100\%$	Good
Yellow	$100\% < \sigma \leq 133.3\%$	Fair
Orange	$133.3\% < \sigma \leq 166.3\%$	Poor
Red	$\sigma \geq 200\%$	Very poor

2. Zoom panel: the nodes are very dense on the graph and it is hard to understand the nature of the nodes layout, the zoom panel helps the users to navigate easily through it and helps to find the relationship among the nodes. In the next section, we explain the zoom in/out functionality in details.
3. Vertical and horizontal scrollbars: these two scrollbars help in exploring and shifting the current view to the right/left or up/down directions and each step represents 1/10 of the graph resolution which is 1000*1000 pixels.
4. Drawing options: this frame provides set of drawing parameters which are used to customize the drawing area. These options include the following functionalities: 1) Control the radius of the node. 2) Transparency, link thickness, zoom in/out, 3) Enable/disable the links drawing or heat map.
5. Searching options: this frame provides set of searching queries to allow user interactively searches for data or relationships in the social network graph. More details in section 6.3.

6.3 DBLP Graph VIS Features:

In this section, we explain in details two of the most important features in our visualization tool which are: graph scaling and graphical search options.

6.3.1 Graph Scaling

This is very important feature that helps in exploring and navigating through the DBLP co-authorship graph due to the limited drawing area which is not large enough to clearly show all the relationships in the graph, which makes the graph so dense especially at the center. Graph scaling gives a clear and close vision inside the social network so the user can study and detect the relationships between the nodes in the graph.

Graph scaling depends on the following factors:

- MDS working area: The geometrical area where the nodes are allowed to move in during the MDS iterations.
- Drawing area: The graphical area where the nodes must be drawn.
- Scale factor: scale factor represents the ratio between the drawing area and a selected region of nodes geometrical area.

Graph scaling pseudo code is show below:

Graph Scaling pseudo code	
$MDS\ working\ area\ (x, y) = 1000 * 1000$ $Drawing\ Area\ (w, h) = 800(width) * 600(height)$ $zoom\ factor\ \ \alpha = 0.5$ $m = \{1\ for\ zoom\ in, -1\ for\ zoom\ out\}$	
1:	Calculate the new zoom value $z \leftarrow \alpha^m . z$
2:	Calculate the shifting value in the width and height: $\Delta w = w_2 - w_1$

$$\Delta h = h_2 - h_1$$

3: Get all nodes $u \in d_G$

4: Calculate the new location of x on the graph

$$x \leftarrow (x - \Delta w).\left(\frac{w}{z}\right)$$

5: Calculate the new location of y on the graph

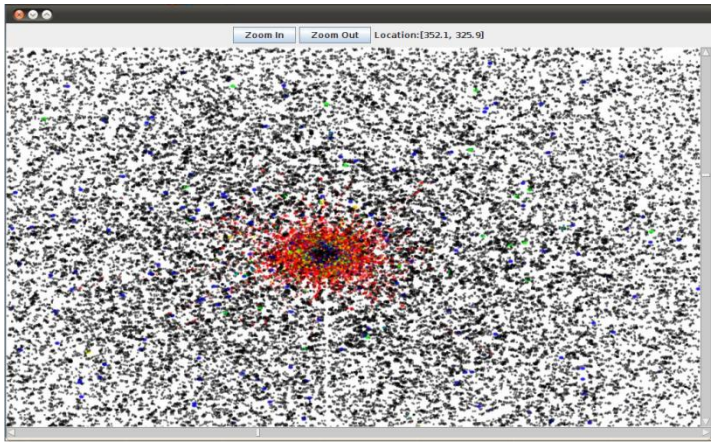
$$y \leftarrow (y - \Delta h).\left(\frac{h}{z}\right)$$

6: Redraw the graph

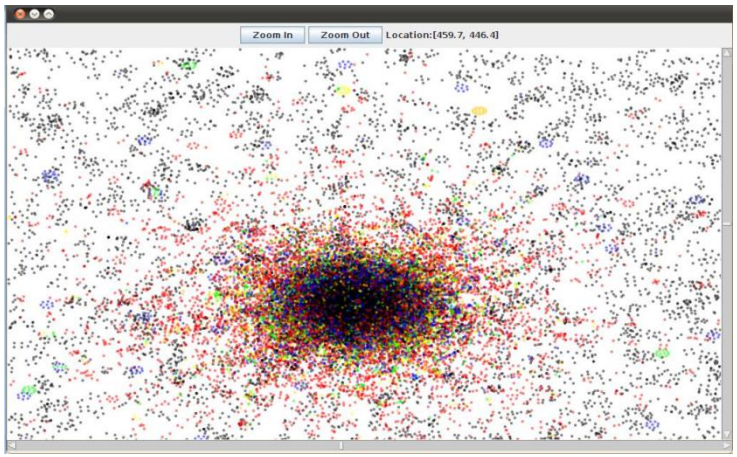
Where m is the zoom mode, and d_G is the next drawing area of graph G.

The best graph layout of the nodes that user can see on drawing area depends on the accuracy of the node location which means that each node must have a high precision value of its (x, y) coordinates, otherwise, nodes with law precision will map to the same pixel during the graph scaling process.

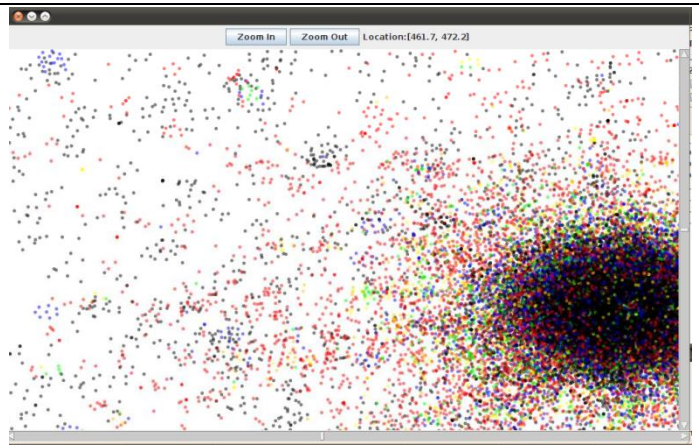
Figure 6.2 shows different views toward the centre of the graph layout during the process of graph scaling (b) shows a close snapshot of the center with link drawing enabled.



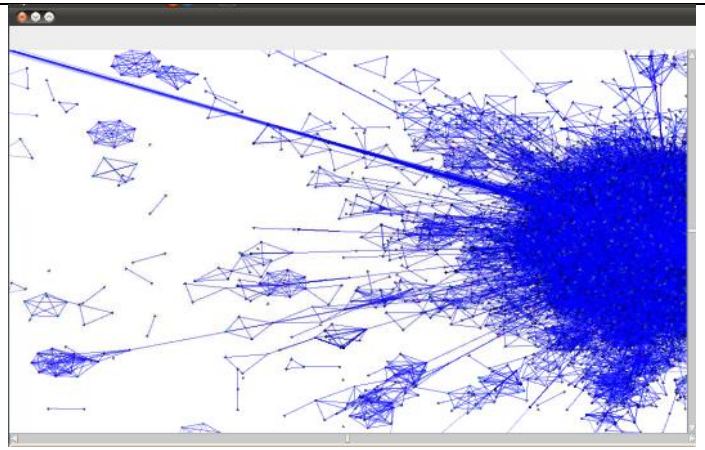
(a)



(b)



(c)



(d)

Figure 6.2 Zooming in toward the graph center

6.3.2 Heat Maps

A heat map is a 2D data visualization technique which uses colors to represent the value of data in colored image. Heat maps show where hot or cold spots are in the data, and enable the user to judge the strength and weaknesses areas in the data. There are many verity types of heat maps such as tree maps, geographic maps, and heat chart maps.

We have implemented a simple heat chart map using java libraries to view where the search results are concentrated. The color of single chart depends on the number of nodes mapped to the same chart area. The color of the chart becomes denser if there are many nodes mapped to it.

Figure 6.3 shows a heat chart map belongs to all authors who have published topics related to sensors research areas, the charts with high dense color represent larger number of related authors than low dense color areas.

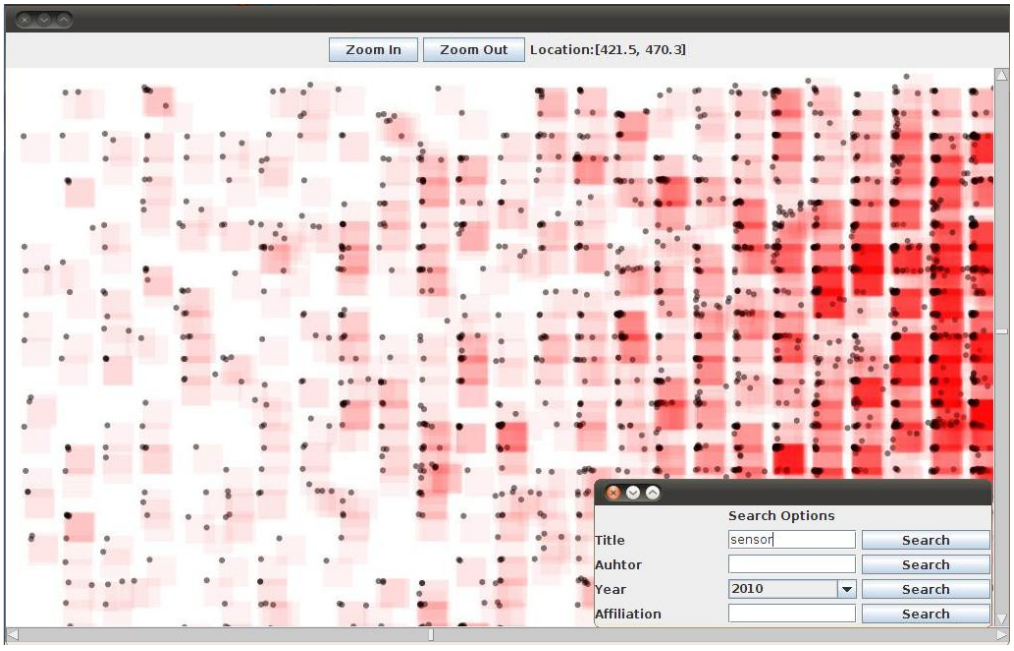


Figure 6.3 Heat map for authors have published topics in sensor research fields

6.3.3 Social Network Visualized Search Result

Searching through graph to find specific nodes on the graph and then visualize the result by drawing the nodes that meet the searching criteria instead of using text, table or charts to view the results. This technique generates a sub-graph contains the search result which aims to better understanding the relationships among the authors and their publications in a visualized virtual world. As said:

"*The drawing shows me at one glance what might be spread over ten pages in a book*"[9].

We provide the user with different methods of searching options in order to retrieve significant information from the DBLP social network graph.

- Search by keyword: this search option gives the user the ability to search for specific keyword repeated in the publication title so that to find any related information between the authors and their publication fields or to see the graph layout for the search result. Also the user can search for more than one keyword at time so that each search result will generate its own graph layout with different coloring code for its nodes. See Figure 6.4 which shows the relationships among authors whose have published topics related to all publications which have MDS keyword in their title.

[9] This quotation is taken from Ivan S. Turgenev's novel Fathers and Sons, 1862.

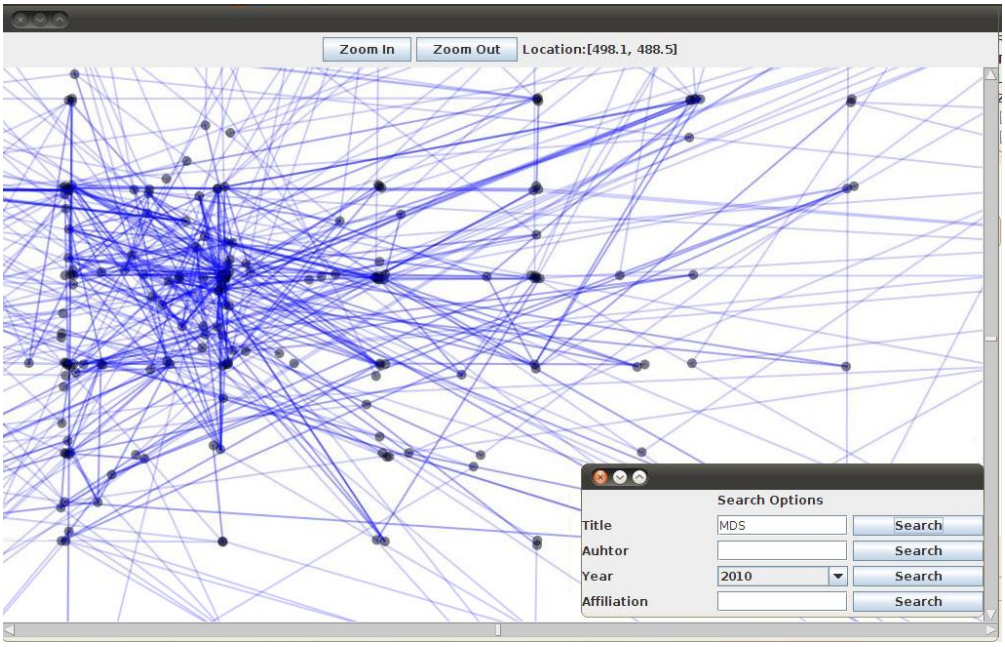


Figure 6.4 The graph layout for related MDS publications

- Search by author name: user can use the author name to find out his relationships with his coauthors, find out their location on the graph, how they are close to each other, and their final layout on the graph. See Figure 6.5 which shows the DBLP graph layout for an author and her coauthors.

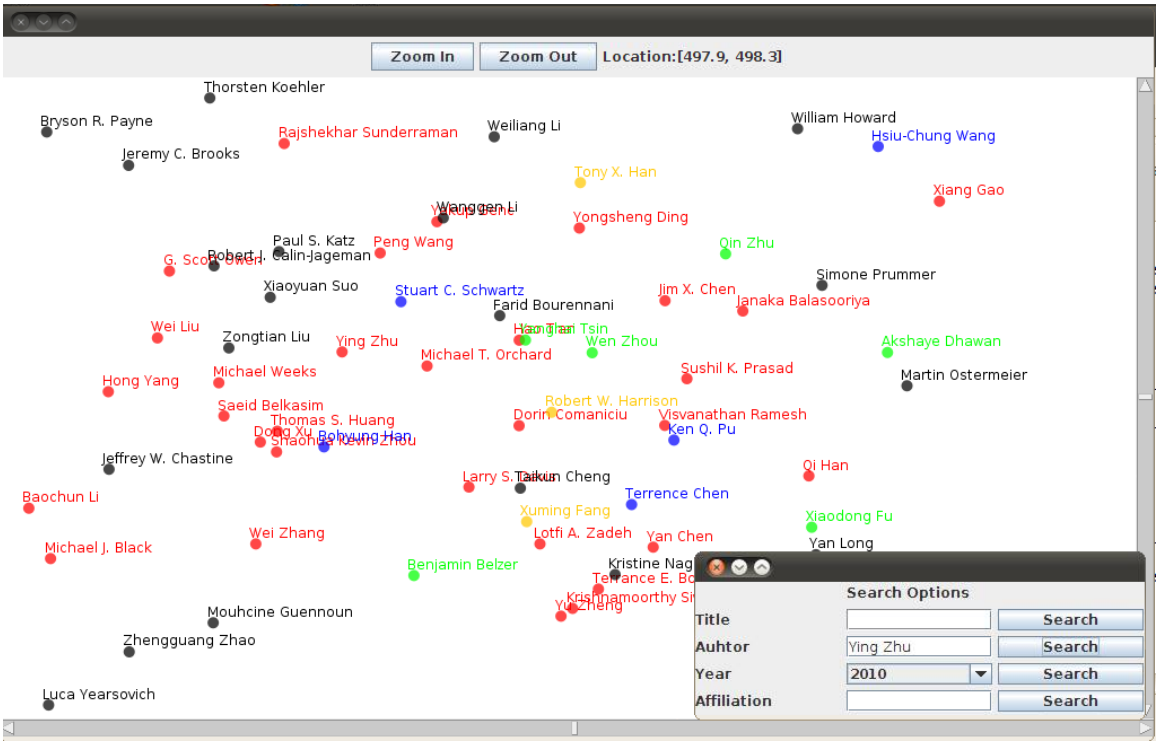
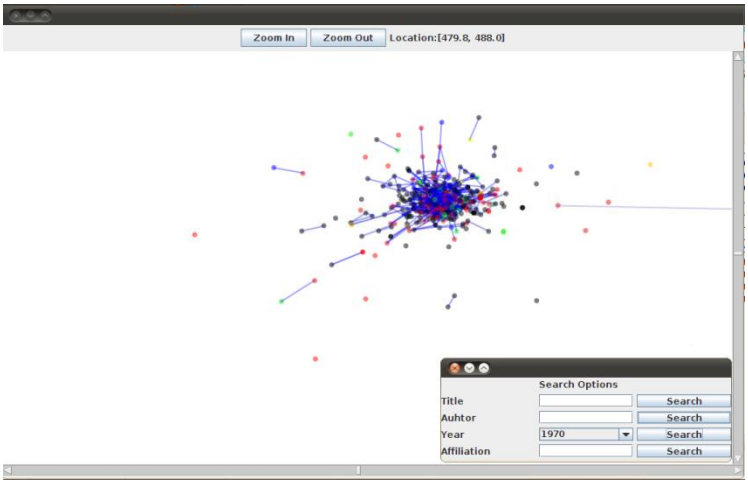
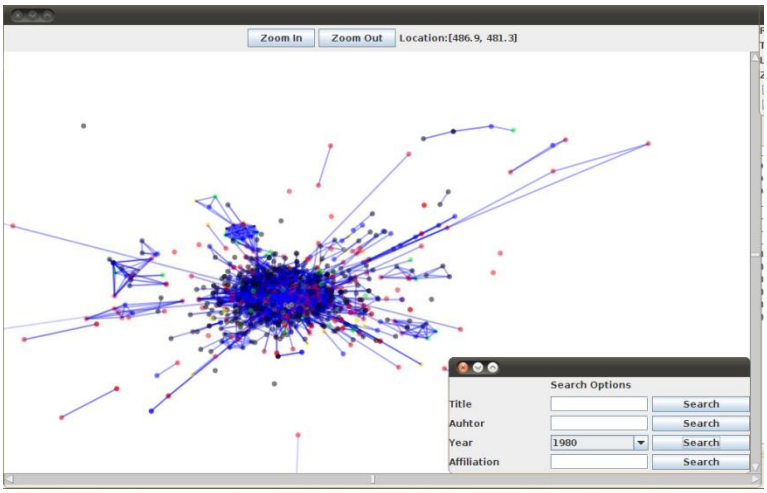


Figure 6.5 The graph layout for an author and her coauthors

- Search by publication year: user can observe the evolution of the graph during the past years since 1950, increasing in number of authors per each year, and number of links which has been added among the authors. Therefore user can observe the increasing in complexity of the author relationships. See Figure 6.6 which shows the difference between the graph layout on year 1970 and on year 1980. Notice that total number of authors increased in 1980 and the relationships among the authors became more complex than in 1970.



(a)



(b)

Figure 6.6 The graph layout in (a) 1970 vs. (b) 1980

- Search by author affiliation: user can use this search to find whether the authors who belong to the same institution/affiliation have collaborated with each other or have worked with other institutions, also user can study the relationships that combine people from different institutions together in the same research areas. See Figure 6.7 which shows the DBLP graph layout for University of Ontario Institute of Technology (UOIT), the graph shows the name of each author above its node.

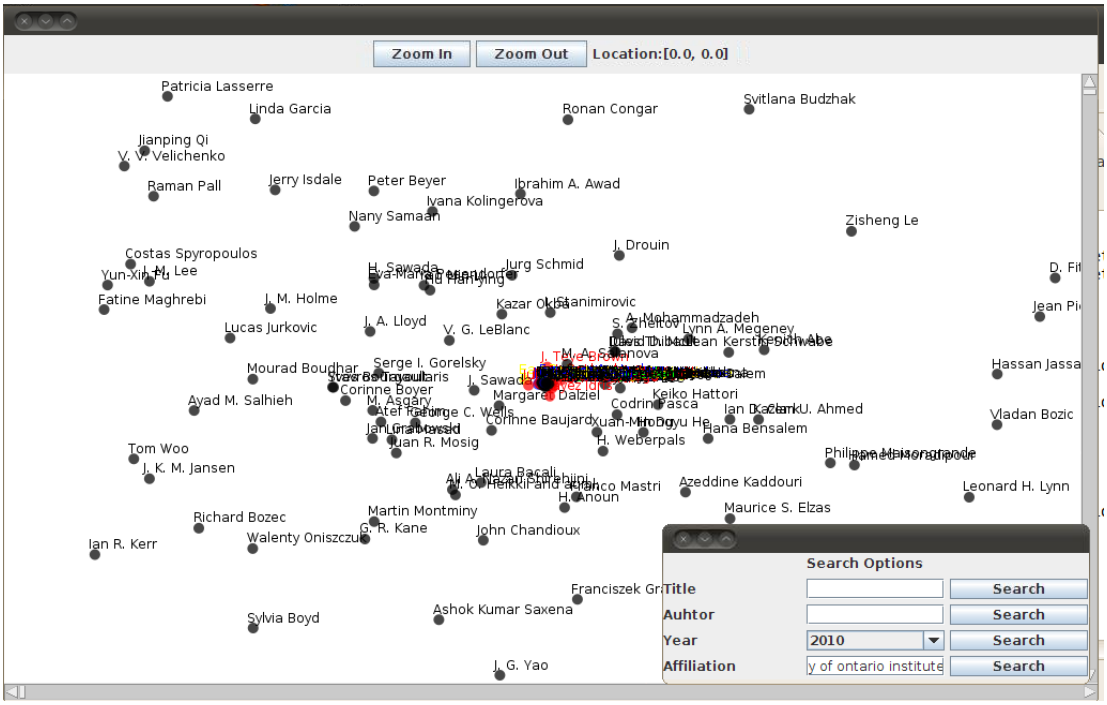


Figure 6.7 The graph layout for University of Ontario Institute of Technology (UOIT)

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Relationships can be found in many fields such as business fields, educational institutions, internet data, human social life, and even in politics. As a result of multiplicity, diversity, and complexity of these relationships, studying and analyzing these relationships became essential, therefore the need to develop different data modules in order to simplify the process of understanding them. These relational data can be described by using words, tables, charts, or graphs, where graphs are considered the most interactively efficient method to view any relational data. Graphs are generated by using mathematical models which describe the relations among its entities, these techniques known as graph theory.

Relationships are usually represented as social networks which contain many entities that are linked together through their respective relations. Many mathematical structure approaches have been developed and proposed to provide the best way to visualize these relations inside the network such as multidimensional scaling (MDS) techniques. Data and social networks visualization has become a hot topic in business fields and computer science research areas because of its aspects in developing a new way to better understand of social relations. Moreover data visualization enables to explore and navigate through data in an easy and interactive way and helps in identifying, classifying, comparing and understanding the relationships in social networks.

In this research, we used DBLP dataset which is an excellent example of a social network dataset. DBLP represents a computer science community which contains a large number of related

authors, publications and venues. In addition it represents a co-authorship social network where authors have multiple relationships among each other such as authors-publications, venues-publications, and authors-venues relationships. Thus, The DBLP co-authorship social network can be used as an experiment to apply and implement data visualization techniques to find the optimal graph layout for huge size social network.

We applied nonmetric MDS algorithm on DBLP dataset to visualize the relationships in 2D dimensions, and measure the similarities of the authors according to the publications they have coauthored and published. MDS output is a 2D spatial configuration of the authors where they are represented on a graph as nodes with links connect them. We found that preparing the proximity adjacency matrix for the nonmetric MDS algorithm in some cases is computationally infeasible because of the large size of DBLP social network. Therefore, we developed a new data structure called Edge Adjacency List in order to store the network graph structure and the proximity data.

Nonmetric MDS iterations aim to minimize the nodes stress by using steepest gradient descent algorithm in order to produce the best graph layout which has optimized stress, and because our proximity measures do not have all shortest paths for all nodes due to the computations complexity of this process, which led to produce a dense graph layouts. As a result, we had many nodes mapped to the same location on the graph, so the next step had been taken was implementing simulated annealing to test if MDS iterations did not stuck in a local minima, and we found that the nature of the graph leads to have this dense area close to the center. Therefore, we proposed and implemented another solution through dividing graph into sub-graphs and then adding a repelling force among these nodes that don’t have a direct link between them.

Three different types of data structure had been implemented in order to enhance the performance of stress optimization and speed up the process of stress convergence. Fist type of data structure

we implemented was static array structure which it is efficient in the term of space and access time, but need to build indexer that references the node ID to its index in the array. In the second type structure, we implemented a hash map to store the graph structure. Hash maps have a constant lookup time and it is efficient in case we have to apply the MDS algorithm on sub-graphs, the drawbacks in using hash map that it consumes a lot of memory space and it is slower in access time than static arrays. The last type we implemented was a custom linked list where we built it using Java. Our linked list was more efficient than the hash maps but less still less that static arrays.

In this research we implemented a multithreaded version of nonmetric MDS in order to improve the overall MDS performance and utilize the two cores we have in our Ubuntu machine. Threads work concurrently on the MDS nodes configuration which had been divided into different ranges so each thread had to work on its region of nodes. As a result, the overall performance of nonmetric MDS improved by 25% when we used two threads.

Finally, we developed a visualization tool for DBLP social network graph which can handle a large number of nodes and links, and it has some important features such as graph scaling which helps the user in navigating through the graph by using zooming in/out functions, drawing options that enable the user to enhance the drawing area, and searching options – such as search by word, author name, year or affiliation - which enable the user to interactively search through the graph and he/she can observe the evolution in the authors relationships and the graph layout over the time, find the collaborations between the universities over the word ,help him/her to find trends or patterns in the relations.

7.2 Summary

In summary, we have proposed a new approach to visualize DBLP co-authorship social network by using nonmetric MDS algorithm which has become as popular method for data visualization purposes. We have built a graph structure for DBLP dataset which has more than 850,000 authors and more than 2.7 million links with weight associated for each link between any two pair of authors, also we have proposed different solutions such as using static array, hash map and linked list to enhance the lookup time of the graph data and improve the runtime for the stress optimization process.

In this research, we couldn't find the full adjacency matrix for all nodes in graph because the huge size of DBLP co-authorship social network. As a result, nodes with strong relationships became very close to each other which led to attract their indirect connected nodes to be close to them on the graph layout. Therefore, we have proposed a new approach to repel these nodes away from each other by integrating a repelling force technique within nonmetric MDS algorithm after dividing the graph into smaller sub-graphs. Repelling force technique allows the nodes to form a much better layout on the graph over the MDS iterations.

We discussed the result of the final graph layout produced by nonmetric MDS algorithm, and we illustrated different types of graphs formed by nodes within the whole large graph, and we tracked the changes occurred in the configuration for selected nodes over the MDS iterations, and we illustrated how the nodes stress was keep converging over the MDS iterations until the nodes reached a final state where they had an optimal graph layout.

Finally, we provided the user with an interactive visualization tool that gives him/her the ability to easily explore and navigate the DBLP social network graph, also we proposed a new searching technique called graph search where the user can graphically search and mine for data and

relationships in DBLP dataset without the need to use the other known classical searching methods.

7.3 Future Work

In this research, we mainly focused on one algorithm in data visualization techniques, for that in the future we can use or integrate other algorithms which can help in reaching an optimal graph layout for the DBLP co-authorship social network. Also we can take the advantages of dividing the DBLP graph into sub-graphs and then find the full proximity matrix for all nodes that belong to the same sub-graph and apply a repelling force among the sub-graphs. In addition, we will use clustering algorithms to find clusters within the graph. On the other hand, we will investigate the collaboration between the institutions over the world depending on their geographical location on the world map and find the relationships among their members. Also we will improve the weight calculation among the authors to be more general where it may be related to number of publication have coauthored, venues they have published in and their quality, and the rank of the institutes they have worked in or graduated from.

References

[1] Min Chen; Ebert, D.; Hagen, H.; Laramee, R.S.; van Liere, R.; Ma, K.-L.; Ribarsky, W.; Scheuermann, G.; Silver, D., "Data, Information, and Knowledge in Visualization", Computer Graphics and Applications, IEEE, vol. 29, no. 1, pp. 12-19, 2009.

[2] France, S. L.; Carroll, J. D., "Two-Way Multidimensional Scaling: A Review", Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 41, no. 5, pp. 644 – 661, 28 October 2010.

[3] The DBLP Computer Science Bibliography, Universität Trier, http://www.informatik.uni-trier.de/~ley/db/

[4] Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze, "Introduction to Information Retrieval", ISBN: 0521865719, Cambridge University Press. 2008.

[5] A ntoine Naud and Włodzisław Duch ,"Visualization of large data sets using MDS combined with LVQ", Department of Informatics, Nicholas Copernicus University, www.phys.uni.torun.pl/kmk.

[6] Michael Ley, "DBLP — Some Lessons Learned", Universit ¨ at Trier, Informatik, Germany, VLDB '09, August 2428,2009, Lyon, France.

[7] Tak Cheung Lam; Jianxun Jason Ding; Jyh-Charn Li, "XML Document Parsing: Operational and Performance Characteristics", IEEE Computer Society, Vol. 41, no. 9, pp. 30-37, 09 September 2008.

[8] Forrest W. Young, "MULTIDIMENSIONAL SCALING", Kotz-Johnson (Ed.) Encyclopedia of Statistical Sciences, vol. 5, 1985.

[9] P. Gill, W. Murray, and M. Wright, "Practical Optimization", Academic Press, London, 1981.

[10] A. H. F. Laender, C. J. P. de Lucena, J. C. Maldonado, E. de Souza e Silva, and N. Ziviani. Assessing the research and education quality of the top Brazilian Computer Science graduate programs. SIGCSE Bulletin, 40(2):135{145, 2008

[11] A.L. Barab´asi, H. Jeong1, Z. N´eda, E. Ravasz1, A. Schubert, T. Vicsek, "Evolution of the social network of scientific collaborations", Physica A: Statistical Mechanics and its Applications, vol. 311, issue 3-4, pp. 590-614, Last revised, February 1, 2008.

[12] A. L. Barab´asi and R. Albert, "Emergence of Scaling in Random Networks", Vol. 286 no. 5439 pp. 509-512 Science 15 October 1999.

[13] Michael A.A. Cox, trevor F. Cox, "Handbook of Data Visualization", PP 316-347, 2008.

[14] Patrick J.F. Groenen, Michel van de Velden, "Multidimensional Scaling", Econometric Institute Report EI 2004-15, April 6, 2004.

[15] Steven M. Holland, "NON-METRIC MULTIDIMENSIONAL SCALING (MDS)", Department of Geology, University of Georgia, Athens, GA 30602-2501

[16] Daniel Tunkelang, "A Numerical Optimization Approach to General Graph Drawing", CMU-CS-98-189, January 1999.

[17] P. Eades, "A Heuristic for Graph Drawing," Congressus Numerantium, vol. 42, pp. 149-160, 1984.

[18] T. Kamada and S. Kawai, "An Algorithm for Drawing General Undirected Graphs," Information Processing Letters, vol. 31, pp. 7-15, 1989.

[19] T. Fruchterman and E. Reingold, "Graph Drawing by Force-Directed Placement," Software—Practice and Experience, vol. 21, no. 11, pp. 1129-1164, 1991.

[20] P. Eades, "Drawing Free Trees," Bulletin of the Institute for Combinatorics and its Applications, vol. 5, pp. 10-36, 1992.

[21] Rodrigues Jr., J F., H. Tong, A J M. Traina, C. Faloutsos, and J. Leskovec, “GMine: A System for Scalable, Interactive Graph Visualization and Mining”, Demo section of the 32nd Intl Conference on Very Large Data Bases, ACM Press, 2006.

[22] Stefan Klink, Michael Ley, Emma Rabbidge, Patrick Reuther, Bernd Walter and Alexander Weber, “Browsing and Visualizing Digital Bibliographic Data”, IEEE TCVG Symposium on Visualization,2004.

[23] P.J. van Laarhoven, and E.H. Aarts, “Simulated Annealing: Theory and Applications”, ISBN-13: 978-9027725134, 1989.

[24] R. Baeza-Yates, B. Ribeiro-Neto. Addison-Wesley, “Modern Information Retrieval”, ISBN-13: 978-0201398298, 1999.

[25] Juan C. Meza, “Steepest Descent”, Wiley Interdisciplinary Reviews: Computational Statistics, 10.1002/wics.117. http://dx.doi.org/10.1002/wics.117. Also available as Lawrence Berkeley National Laboratory Technical Report, LBNL-3395E, 2010.

[26] Cay S. Horstmann and Gary Cornell, “Code Java”, Chapter 1: Multithreading, page 02-84, ISBN-13: 978-0201398298, 2008.

[27] Bradley Huffaker, Evi Nemeth, k claffy, " Otter: A general-purpose network visualization tool", CAIDA: Cooperative Association for Internet Data Analysis, 1999.