San Jose State University
SJSU ScholarWorks

Master's Projects

Master's Theses and Graduate Research

2008

Online visualization of bibliography Using Visualization Techniques

Bharath Kumar Manur Venkataramana
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the Computer Sciences Commons

Recommended Citation

CS298 PROJECT REPORT

Online visualization of bibliography Using Visualization Techniques

Subject	CS298	Report	CS 298 Project Report
Project Advisor	Dr. SoonTee Teoh	Semester	Spring 2008

First Name	Bharath Kumar	Last Name	Manur Venkataramana

ACKNOWLEDGEMENTS

ABSTRACT

Visualization is a concept where we can represent some raw data in the form of graphs, images, charts, etc. which will be very helpful for the end-user to correlate and be able to understand the relationships between the data elements in a single screen. Representing the bibliographic information of the computer science journals and proceedings using Visualization technique would help user choose a particular author and navigate through the hierarchy and find out what papers the author has published, the keywords of the papers, what papers cite them, the co-authors along with the main author, and how many papers are published by the author selected by the user and so on in a single page. These information is right now present in a scattered manner and the user has to search on websites like Google Scholar [1], Cite Seer [2] to get these bibliographic records. By the use of visualization techniques, all the information can be accessed on a single page by having a graph like points on the page, where the user can search for a particular author and the author and its co-authors are represented in the form of points.

The goal of this project is to enhance current bibliography web services with an intuitive interactive visualization interface and to improve user understanding and conceptualization. In this project, we develop a simple web-interface which will take a search query from the user and find the related information like author's name, the co-authors, number of papers published by him, related keywords, citations referred etc. The project uses the bibliographic records which are available as XML files from the Citeseer database[2], extracts the data into the database and then queries the database for the results using a web service. The data which is extracted is then presented visually to allow the user to conceptualize the results in a better way and help him/her find the articles of interest with utmost ease. In addition the user can interactively navigate the visual results to get more information about any of the article or the author displayed. So here we present both paper centric view and author centric view to the user by representing data in terms of graphs. The nodes in the graphs obtained for paper centric views and author centric views are color coded based on the paper's weight parameter (popularity of the paper). For the paper centric view, the papers which are referring other papers are represented by providing a directed arrow from referred paper to referenced paper. Overall the idea here was to represent this related data in the form of a tree, so that the user can correlate all the data and get the relationships between them.

TABLE OF CONTENTS

LIST OF FIGURES

1. <u>INTRODUCTION</u>

Data Visualization involves data mining at the backend and Visualize the data which we extract from the data mining approach onto the frontend. The Data mining part here involves extracting the data using some clustering methods and find related data from the web and store that in the database. This part is very important in order to visualize these related data in the future. After extracting this data, we can represent this data in the form of graphs, charts, Images, etc.. by applying some Visualization algorithm to place the nodes in the graph or points in the charts on the screen, with the data loaded.

This method of representing data is now used in the field of business, representing stocks related data in the form of charts, social networking websites and many more. Some of the examples where this method is used are in Amazon[3], where the user can search for a particular product and when he/she clicks on that product, it will redirect to another web page, where he/she will be able to view a graph kind of structure (tree-like), which will have related products that are associated with this product.

Another place where this methodology is used is in Twitter[4], which is a social networking website which will allow the user to find his related friends and the next closest friends, who may be in your other friend's list. This type of data is represented in the form of a graph, where the Photo of one user is connected to many other Photos of its friends and their related friends and so on in the form of a tree. By having this whole information on one page, we will be able to zoom in to which friend we are interested in knowing and get more information on that. In both the examples, the respective web service is used to load the data onto the graph.

For placing these tree-like structures we need to use some Visualization techniques to position the nodes of the graph in the layout. The way the graph looks is based on the technique we use. In this project, Force-Based Directed Layout Algorithm [5] is used to position the nodes and edges. But many other techniques also are very useful in representing the graphs. One of the key points to look out while implementing these techniques are how flexible are those techniques when we have to make it customizable for different requirements. For example, if we have to use a visualization technique to implement a Visualization Graph for a big Family providing the relationships between family members upto 6-7 levels. Then we have to make use of

Hierarchical Visualization Algorithm [6], which will position the nodes in a proper ancestor-descendent format. So it's important to know which technique to use for the type of implementation you are looking for.

Figure 1.1 Twitter Graph Visualization [7]

In this project we use the idea of the Graph Visualization concept explained above to represent the Bibliographic data such that we can establish relationship with authors, papers, related keywords and so on. Here we use Citeseer[2] database from the web and get the bibliographic records from that and store it in the database. By using some clustering algorithms, we will get the related data using a web service, which is in the form of an XML file and then we parse the XML file and apply a Visualization algorithm to represent the data on the web page.

Here we have a search result kind of format, where we search for papers and get related papers based on the keyword provided in the search. We also provide a new search for authors

and get related authors based on the search results. We will use this search results and click on one of the paper and get a paper-centric view, which will give us a tree-like structure showing the Paper as the main focus and the papers referenced by this paper, the authors of the papers and some of the papers the authors have written. The user can navigate through the hierarchy by navigating from first level to next and so on. The data here is loaded dynamically as and when the user needs more data. The authors and their related paper data nodes in the graph is of less significance, so we reduce the size of the nodes of this data. Also the papers here are color-coded based on the weight associated with the paper, which is referring to the popularity the paper has received.

For the author centric view, the main author and their related co-authors are represented using the tree-like structure showing the main author as the focus and each author will be having their papers and which is linked to the respective authors in the graphs. The author's which have written papers together are shown closer by providing the bonding between them. Here also the user can navigate from level one to next and so on. Here the papers written by the authors are of less significance, so we reduce the size of the nodes of this data. Also the authors are color-coded based on the weight parameter, indicating the popularity of the author.

In this project, the main focus is on getting an online Visualization for the bibliographic data. As seen right now in the internet world, there is so much irrelevant data and some very much related data. To relate to those data which are related to each other, we need to get a Visualization chart. Without that it's very difficult to collect all the data and relate it separately. In this project, we eliminate this concern by providing a Visualization Graph for the bibliographic data and help the user to get all the related data of authors and papers on a single screen. This would eliminate that concern of finding for information on different websites and refer different papers and get the related information.

2. <u>OBJECTIVE</u>

The objective in this project is to create Graph Visualizations for Bibliographic Records from Cite seer [2] and represent these Graphs on a web browser. The main goal in this project is to create a search engine for the bibliographic papers and authors, along with that create Paper-Centric and Author-Centric Data Visualization Graphs which will provide all the details about the authors, papers, related keywords, co-authors and many more.

3. <u>PROBLEM STATEMENT</u>

In this project, the main challenge is to find a suitable Visualization Algorithm for drawing a Graph on the web page and load the data dynamically onto the graph nodes. So in this project we need to use some Clustering techniques to get related data from the Cite seer[2] database. After getting the related data we need to use that data in the form of web service and we need to parse that data from the web service and feed the data to the Visualization Tool.

After parsing the data we need to convert that into some XML format in order for the Visualization tool to recognize. The XML data should now be used by the Visualization Tool which will create the data nodes on the web layout and also link the nodes according to the data provided. The nodes in the web layout have to be positioned such that they will all be displayed on the web layout in a proper manner. For doing this we need to use some Graph Visualization Algorithm which will calculate the position of the nodes and many more parameters and position the nodes along with the links on the web page.

By using these analogies we need to create 1) Author-Centric and 2) Paper-Centric Views for the Bibliographic records. The Graph representation should highlight all the data that the user will be looking for in a single screen and should be able to correlate the data and establish a relationship between the authors and papers.

4. <u>FLEX VISUALIZATION TOOL</u>

Flex [8] is a useful Visualization tool that is used to represent related data in a graphical format in the form of charts, graphs, tables, etc.. Flex is like a presentation server which is used to develop Enterprise Rich internet applications (RIA's) [9]. This presentation server is installed on a J2EE application server, along with several useful user interface components like SpringGraph [9]. Along with this, Flex has a very useful object-oriented support which is very useful in interacting with the database objects and also provides very good user interactions with the real-world applications. It also supports XML-based data and is very useful in populating the data from the XML onto the user interface components very easily. This is one of the important features that were very useful in this project. [9]

Flex provides a very good development atmosphere for the developers with runtime support and help them develop very good frontend applications that interact with the database very effectively. It also takes advantage of using Macromedia Flash Player [10]. This Flash Player is very useful when we run the Flex application; this Flash player will interact with the databases, web service, objects created in the flex application and so on. This player is also useful in debugging the Flex application in runtime.

Another important feature present in Flex is its ability to bind the data variables, so that the data in one object can tie to the other object. For example, in Flex there is a component called HSlider and another component called Text, so we can copy the value provided in HSlider component onto Text component using the id variable in the component. This process is called binding the data variable in HSlider to Text object. This feature is very useful as we can declare

the data binding variable as public, so that it is very useful in binding the data value to that variable throughout the application.
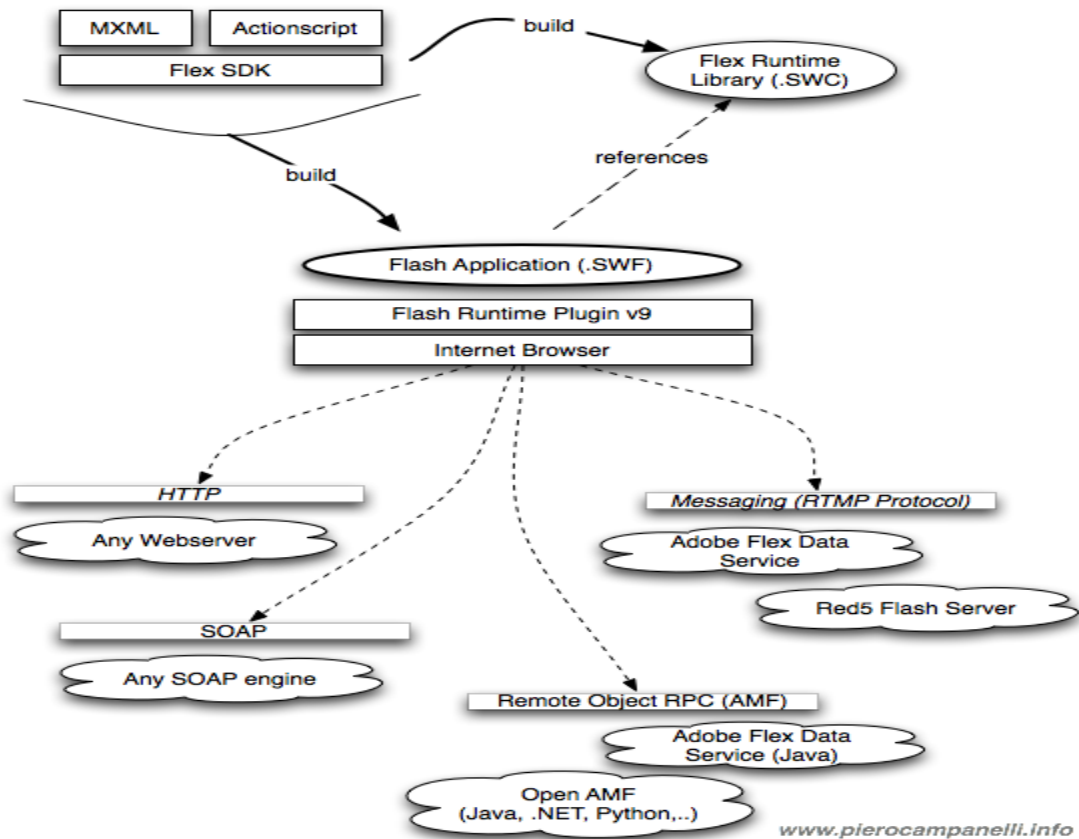
Figure 4.1   Flex Architecture [11]

The Flex application will use MXML applications for creating a controller and then use ActionScripts for View and Model. These applications are built using a Flex SDK into a Flash Application in .swf format. The Flash application will reference Flex Runtime Library which will be having some standard libraries and customized libraries. The customized libraries are usually built using the same MXML and ActionScripts. The Flash Runtime Plugin is used for debugging purposes which will help us debug the program by setting some breakpoints in the program using "trace" function which is similar to "printf" function in C.

The web browser will also do a Remote Procedure Call using the Adobe Flex data service inorder to get the data when a web service functionality is implemented in the Action Script. Flex will also support other protocols like HTTP, SOAP and RTMP. [11]

How to create Graph Visualization using Flex?

By using SpringGraph [12] component in Flex we will be able to display the items in such a way that we will able to see related items in the form of a tree. Each item will be having its sub items or related information which can be viewed by navigating through the tree. Here the items that are related are linked by having a line connecting them. The links and the nodes for these items are drawn using a Visualization algorithm which will calculate the layout size and will accordingly position the items on the screen. This component will also help us to scroll over the graph, expand the items and also helps us in zooming in and out the graph by providing the contracting and expanding effect. So this will help the user to talk to individual items and also get more information about it.

The data that is provided for getting this graph is in the form of XML which is fed to the Spring Graph component which will use this as a data provider for that particular node and load the data to that node in the graph. One of the examples where this component is used are:-

1) Amazon items:- In this application, they call Amazon web service to get the dynamic XML and then they parse the XML data and load the data onto the Spring Graph component. As shown in the figure, there are many Apple Ipod products which have related items linked with each other. The next level items have their related items and so on. So the user can have a full set of items and look at the description of all the items and their ranks or reviews on a single screen and choose what he/she wants to buy.

Figure 4.2 Amazon Item Visualization [13]

The key to get a very good graph structure to be displayed in the web browser in this project is due to the use of Spring Graphs in flex. SpringGraph[12] is the main component - you give it a graph of items and links, and specify the Flex UI for displaying each item. The graph can be given as XML, or built from the Graph class. The component follows the data Provider/item Renderer pattern seen in other Flex components such as DataGrid or List, with one difference - the data Provider is a graph rather than an array.

Here the Spring Graph [12] component is used to display the author centric view and paper centric view graphs on the web browser. User can click on the nodes in the browser and can navigate through the similar items dynamically on the web page. For example, if we have a list of authors linked together and if we click on one of the co-author in the graph, we should get the tree structure for that particular co-author (meaning we should get the respective co-author related data). So the graph tree should keep on getting updated dynamically.

Overall Flex provides a very useful interaction with the web service and the flex objects created, as this kind of feature is very useful in real-time applications where we can develop rich front-end interfaces and as well have very good web service and database support for that applications. [12] One of the main drawbacks while building Flex application is we need to read the XML data we get from the web service as a string and then parse the data and then again reformat it to the XML format and read again. This is an overhead and can slow down the

application in some situations, where the user is making a lot of queries to the database, so each time the user makes a query, a web service is created and the Flex application has to parse it and reformat it, which will take some time.

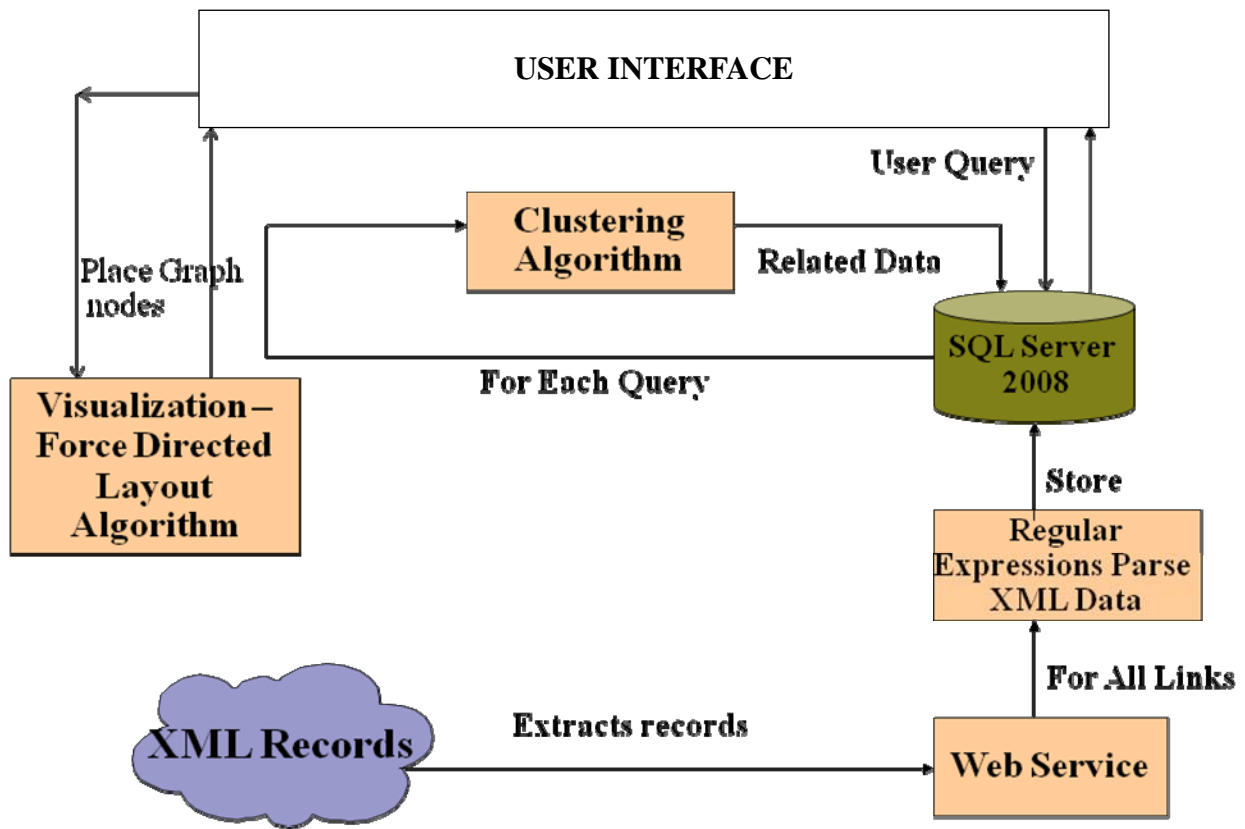# 5. <u>ARCHITECTURE</u>

<u>Figure 5.1 – High-Level Architecture</u>

The user can enter the search query on the User Interface and will get the related information by querying the SQL database and then running a clustering algorithm on the search query and display that data on the web browser in the form of datagrids using flex. Also the user can click on one of the papers or authors got from the search query and get author centric view or paper centric view for that particular author or paper respectively. The author centric view will display the author name and the related co-authors in the form of graphs, by linking the main author with its co-authors and also displaying the papers published by the main author. In case of paper centric view, the main paper is displayed along with the referenced papers by the main paper along with the title and authors for each of the papers on the screen. All the display is done

in graph format. So user can click on the authors/papers in the graph tree and can get similar related information by expanding the graph for the related information on the screen.

The XML records for this Bibliographic data are parsed using XML parser and stored in the SQL server database. A clustering algorithm is applied on the data based on the search query provided by the user. After getting the results from the search query through the webservice in the form of XML file, we read the XML file through Flex Visualization tool and extract the related information from it and sketch a graph structure based on the contents. The graph structure is built using Spring Graphs [12] in flex. The results we get from the search query we also display them in the datagrid using flex on the web browser. So user can click on any paper or author and get its corresponding centric/detail views in the form of graphs.

6. <u>SAMPLE XML STRUCTURES</u>

PAPER-CENTRIC VIEW XML STRUCTURE

```xml
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://biblio.cs.sjsu.edu/">
<paper>
  <id>978</id>
  <title>Software Transactional Memory</title>
  <description>DESCRIPTION...</description>
  <weight>0</weight>
<authors>
<author>
  <id>3022</id>
  <name>Nir Shavit</name>
  <weight>8</weight>
    </author>
<author>
  <id>3023</id>
  <name>Dan Touitou</name>
  <weight>2</weight>
    </author>
    </authors>
<references>
<paper>
  <id>14421</id>
  <title>The MIT Alewife Machine: A Large-Scale Distributed-Memory
    Multiprocessor</title>
  <description> des...</description>
  <weight>32</weight>
<authors>
```

```xml
<id>3554</id>
<id>40634</id>
<id>41473</id>
<id>25008</id>
<id>5983</id>
<id>41474</id>
<id>3553</id>
<id>41475</id>
<id>38473</id>
  </authors>
  </paper>
  </references>
<referredby />
  </paper>
  </root>
```

7. <u>VISUALIZATION ALGORITHMS</u>

Visualization algorithms play a very important role in placing the nodes in the graph along the layout. This is very helpful as we have to represent the data on the User Interface in the form of points/ nodes and spread those nodes along the layout in a certain manner. Without doing this, the nodes will be placed one above the other and the user will not be able to view the graph properly. The relationship between the data nodes will be clearer when we use these algorithms. Some of the existing Visualization algorithms are:-

1) **Force Based Layout:-[5]** In these algorithms, electric and magnetic forces are applied on the nodes and edges between nodes.

2) **Orthogonal Layout** [14]:- Here the nodes and edges are drawn on the layout by placing the edges horizontally and vertically. This will reduce edge crossovers and also will increase the area covered.

3) **Tree Layout** [14]:- These algorithms help us to create a tree-like structure with a root node and leaf nodes kind of structure. But these class of algorithms will not provide graphs which have cycles.

4) **Hierarchical Layout** [14]:- These kind of algorithms provide a ancestor-descendent relationship between the node objects and is very useful when we have to represent data of a big family in the form of graphs.

FORCE DIRECTED LAYOUT ALGORITHM [15]

Force-based directed algorithms are useful for drawing graphs in a very nice manner. The main purpose of this algorithm is to place the nodes of the graph in the layout such that all the edges are of same size and the overall graph having less crossing edges.

In this algorithm, the overall layout is considered as one physical system, with the graph nodes acting as bodies of the whole system. [15] There will be some kind of forces between these nodes like an electric force or magnetic force. So the nodes here are connected with each other in the layout through some force parameter. The force parameter between two nodes is always directly proportional to the distance between the two nodes. [15]

Hooke's Law is used to calculate the forces surrounding each node and then establish a state of equilibrium among the nodes. [15]
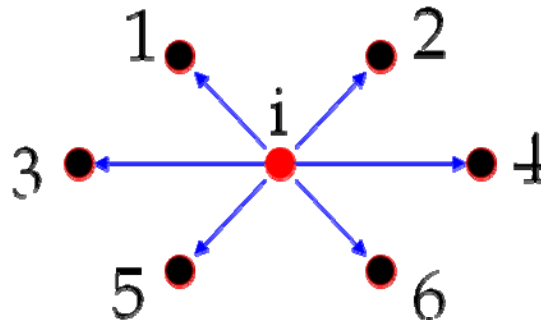

Figure 7.1 Force on Graph Nodes [15]

The basic logic behind this algorithm is if the two nodes are not linked with an edge, then a repulsive force is applied to move the nodes away from each other. This is done because the two data nodes are not related to each other. If there is an edge connecting two nodes, then an electric attractive force is applied so that the two nodes are closer to each other. This means that the two nodes are related to each other. This way the nodes in the graph will be positioned based on whether there is relationship between the data contents of the nodes or not. [15]

The Force calculation for this algorithm is shown as below:-

$$F = k \cdot d$$

$$F_i = \sum_j k_{ij} \cdot d_{ij}$$

$$\sum_j k_{ij} \cdot (x_j^{New} - x_i^{New}) = 0$$

$$\sum_j k_{ij} \cdot (y_j^{New} - y_i^{New}) = 0$$

$$x_i^{New} = \frac{\sum_j k_{ij} \cdot x_j}{\sum_j k_{ij}}$$

$$y_i^{New} = \frac{\sum_j k_{ij} \cdot y_j}{\sum_j k_{ij}}$$

Figure 7.2 Force Calculations [15]

Basic Pseudo – Code for this algorithm:-

1) Compute the initial layout for positioning the nodes.
2) Repeat
3) Compute the forces between the edges.
4) Construct a geometrical cluster
5) For each node N in the graph
6) Compute the non-egde forces on nodes N.
7) End forloop
8) Move the nodes based on repulsion factor.
9) Update the boundary area.
10) Proced until stopping condition.
11) End. [15]

Advantages of Force-Directed Layout algorithms:-

- ➢ Shows very good results for graphs with size 50-100 nodes.
- ➢ The graph layout is very much simplified, so that the graph is easy to follow.
- ➢ It helps in building a very good graph-based interactive system. [15]
- ➢ These graphs are highly flexible and we can add more additional properties to it as and when needed.
- ➢ These graphs act like spring forces and can be stretched or converged like a rubber-band.

Disadvantages of Force-Directed Layout Algorithms:-

- ➢ It doesn't support graphs of larger size (more than 100 nodes).
- ➢ The complexity of this algorithm is $O(n^2)$, which increases the cost as it is used at each step. [15]

Examples of Force Directed Layout Algorithms:-
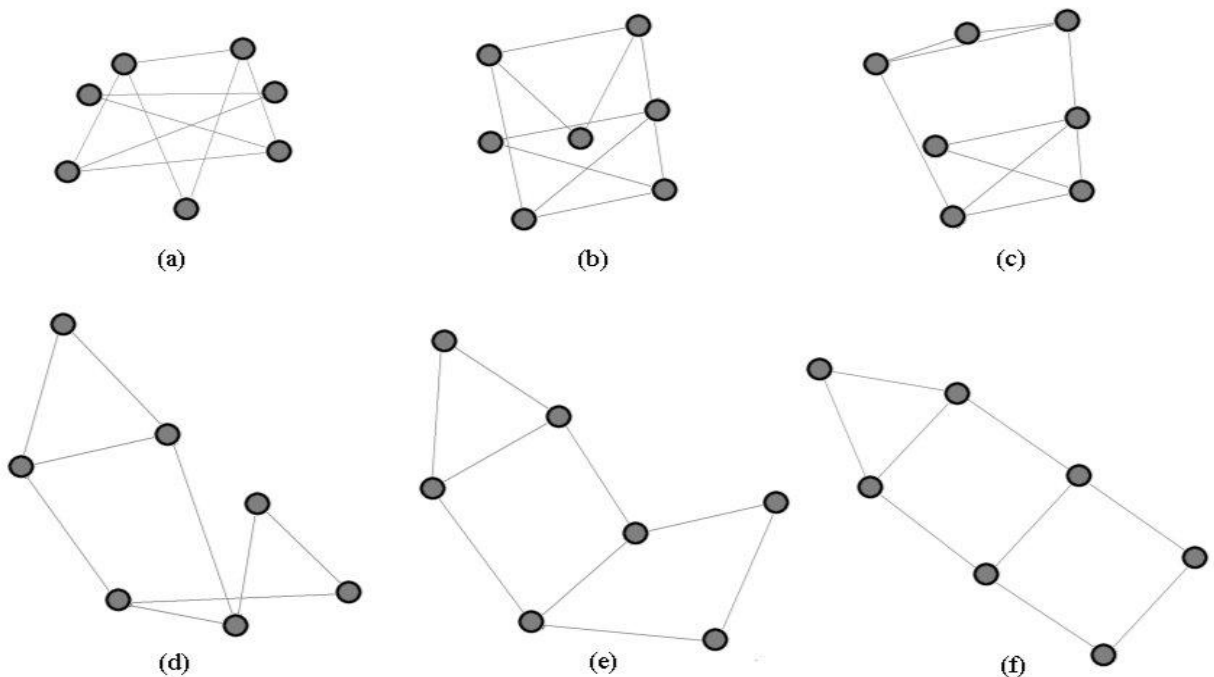
Figure 7.3 Force-Directed Algorithm applied on the Graph nodes after a series of Steps[15]
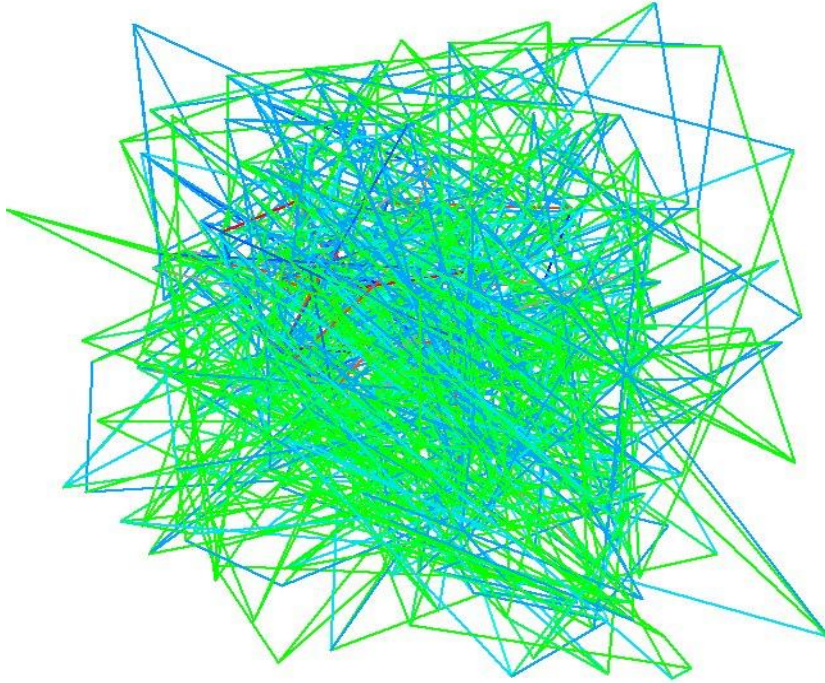
Figure 7.4 Force Directed Algorithm applied on many nodes [15]

The two figures above show how this algorithm is applied to the undirected graphs and the result of this algorithm after running for a series of iterations. The algorithm is run for some steps until the equilibrium state is found. This prevents the crossing over of edges in the graph to some extent. As seen above the nodes which have some relation are positioned close to each other and ones which have no relation are far apart. The clear distinction of the same is seen after a series of steps. Overall these classes of algorithms are very useful for positioning the nodes around the layout.

Clustering Algorithm

The clustering algorithm will cluster the data based on the search query provided by the user. If the search query is the name of the author, for example the search query is John, then the clustering algorithm will extract all the authors whose name has John as one of the string in their names. Also it will extract the related papers published by that authors and the bonding information of that author with its co-authors. In case of a paper search query, the clustering

algorithm will extract all related papers which have any string which matches the search query string and will also get the related papers referenced by that paper.

Author Centric approach provides details about the author like his name, papers published by him and the related co-authors. Also a weight parameter is calculated for each author based on how many papers the authors have written and so on.

Paper Centric approach provides details about the paper name, description, the citations of other papers by this paper and so on. Also a weight parameter is calculated for each paper based on how many papers refer this paper (popularity factor of this paper).

8. <u>SPRING GRAPH USING FLEX</u>

What is meant by a Spring Graph?

Spring Graph[12] is a component in Flex, which helps us to realize Graph visualizations. This component helps us to create an interactive system by dynamically loading the data to the Graph nodes, so that we are able to develop the Tree-Like structure. By this way, we can draw a link between two nodes, if they are related to each other using this component. We can also define a repulsion factor in this component, which will tell how much the nodes should be pushed away from each other.

Why is this component useful?

This component plays a very important role when we have to show the relationship between related data in the form of graph. For example, in case of Social networking websites like Twitter[4], it is very easy provide a relationship between me and my first level friends and second-level friends and so on, using the form of a Tree-Like structure. In this project, we use this component to get related data like authors, co-authors relationship, papers and referenced papers relationships and so on and helps us to create a graph structure and allows us to interact between the second-level tree and so on.

Uses of Spring Graphs in Real-World Applications:-

Example 1:- Amazon[3] uses Spring Graphs to update information about the related products and create a graph visualization.

Example 2:- Twitter[4] uses Spring Graphs to update information about the related Users and create a graph visualization.

How Spring Graph is useful in this Project?

Spring Graph[12] is used to create a Graph named Paper and Author and add nodes to these graphs and link them based on the information provided in the xml. By using this component, we are able to create this interactive system, which is so popular in the area of data visualization now.

The Spring Graph is created using Graph: Paper = new Graph( );

Adding a new node to this graph named "Paper" is – Paper.add( );

Linking a node to this graph is done by getting the id of the node in the graph, the present node has to be linked to. Paper.link(newNode, LinkTo); [12]

The link feature has properties like- setting the line style, thickness of the line, also for the paper referencing another paper, an arrow is drawn indicating the reference. The arrow function is implemented inside the Spring Graph component when the edges are drawn by the component.

We can also remove the links between items by using Paper.remove(newNode, LinkTo). We can set the repulsion factor between the nodes by providing the repulsion factor. Also autofit feature can be enabled by setting a Boolean value inside the program, specifying true or false. The data is loaded to this component using the xml format with the node created first as the root of the graph and the other nodes are linked accordingly based on whether they are related to each other or not.

9. <u>IMPLEMENTATION DETAILS</u>

MVC architecture:-

The Model View Controller concept is used in implementing this visualization. Here for Paper-Centric approach, PaperDemo.mxml – file is the controller, which will pass the control to the Model – Paper Item , which will do a call to the Paper webservice and gets the data loaded and parse the data and convert it into a standard xml format. Then the controller makes a call to the item renderer class, which is the PaperItemView – View class, where we use the Vertical and Horizontal Boxes for the nodes and coloring the nodes and so on.

Input and Output to the Program:-

For the paper-centric view, the input to this algorithm is :-

1) Create a graph , for example named – Paper = new Graph();
2) Get the paper ID and get the data using the ID by using a webservice call to the database and format the data to an xml List format, so it can be read properly.
3) After getting the data, create a new item node to the graph labeled, the paper id and the paper's title by – Paper.add(item).
4) Then after adding the new item to the graph, check whether the paper has any other referenced papers which can be linked to it.
5) If there is any papers referencing this main paper, link those papers to the Main Paper Item by – Paper.link(newitem, item). If there is no other papers referenced, then that's the end of the graph.
6) For each paper item in the graph, create new author items, like the authors who have written that paper by using the Paper.add and Paper.link features.
7) Also get the weight factor from the XML data and color the Paper Items based on the popularity of the paper. [12]

For the author-centric view, the input to this algorithm is :-

1) Create a graph , for example named – Author= new Graph();

2) Get the author ID and get the data using the ID by using a webservice call to the database and format the data to an xml List format, so it can be read properly.

3) After getting the data, create a new item node to the graph labeled, the author id and the author's name by – Author.add(item).

4) Then after adding the new item to the graph, check whether the author has any co-authors which can be linked to it.

5) If there is any co-authors referring to this main author, link those co-authors to the Main Author Item by – Author.link(newitem, item). If there is no other co-authors, then that's the end of the graph.

6) For each Author item in the graph, create new Paper items, like the Top 2-3 Papers that have been written by the authors by using the Author.add and Author.link features.

7) Also get the weight factor from the XML data and color the Author Items based on the popularity of the Author. [12]

Parsing:- The data which we get through the webservice has to be parsed by using some regular expressions and make the data formatted in a xml format, so that its easy to load the data to the graph nodes.

How the Force Directed Layout Algorithm is used in the implementation?

This algorithm is useful in placing the spring graph component nodes in the layout by applying electric and magnetic forces. This is done by applying a repulsive magnetic force between 2 nodes by varying the x and y axis distances. Then if the 2 nodes are not linked, the applied magnetic repulsive force is little more, so that they are placed little away from each other.

Now if the 2 nodes are linked, then the edge between them is applied an attractive electric force on the x and y axis, so that these 2 nodes attract towards each other. This analogy is based on the fact that distance is directly proportional to Force applied, $F = d * v$ [15]. By varying the

distance, the force is varied. This makes the Graph edges look stretchy, like a rubber band, which will help the user to drag the nodes on the layout and place them anywhere he wants to on the screen.

Main Modules used in the Project:-

Paper Centric View Modules:-

1) **PaperSearchQuery.mxml :-** In this module, the papers are queried from the database and this will display the search results in the Flex data grid. The user can click on any of the search result, so that he/she will be able to navigate to the Paper centric view of the selected Paper.

2) **PaperDemo.mxml :-** This application will take the selected paper id from the PaperSearchQuery and will use that Paper Id to load the main paper that is selected to get the Paper Centric View graph. This module also implements the Zoom in and Zoom out feature. It also has recursive query calls to the web service in order to load the data in the second-level and so on. This module acts like a controller n the MVC framework. This creates a new Model called PaperItem, which will get the actual data from the web service. This module also embeds the item renderer – PaperItemView, which is the View in this MVC framework.

3) **PaperItem.as :-** This is an action script, which calls the Paper web service and gets the xml file and parses it by using some regular expressions and convert that in to a proper xml format. Then this will return the data to the controller.

4) **PaperService.as :-** This is an action script, which calls Paper web service by using loader request and response methods and gets the xml formatted data from the web service.

5) **PaperItemView.mxml :-** This is the module that will actually create the Node in the front-end by using Vertical boxes and Horizontal Boxes and also implements the color coding of the nodes based on weight parameter got from the PaperItem Model.

Author Centric View Modules:-

1) **UserSearchQuery.mxml :-** In this module, the authors are queried from the database and this will display the search results in the Flex data grid. The user can click on any of the search result, so that he/she will be able to navigate to the Author centric view of the selected Author.

2) **AuthorDemo.mxml :-** This application will take the selected paper id from the AuthorSearchQuery and will use that Author Id to load the main Author that is selected to get the Author Centric View graph. This module also implements the Zoom in and Zoom out feature. It also has recursive query calls to the web service in order to load the data in the second-level and so on. This module acts like a controller n the MVC framework. This creates a new Model called AuthorItem, which will get the actual data from the web service. This module also embeds the item renderer – AuthorItemView, which is the View in this MVC framework.

3) **AuthorItem.as :-** This is an action script, which calls the Author webservice and gets the xml file and parses it by using some regular expressions and convert that in to a proper xml format. Then this will return the data to the controller.

4) **AuthorService.as :-** This is an action script, which calls Author web service by using loader request and response methods and gets the xml formatted data from the webservice.

5) **AuthorItemView.mxml :-** This is the module that will actually create the Node in the front-end by using Vertical boxes and Horizontal Boxes and also implements the color coding of the nodes based on weight parameter got from the AuthorItem Model.

10.   SNAPSHOTS
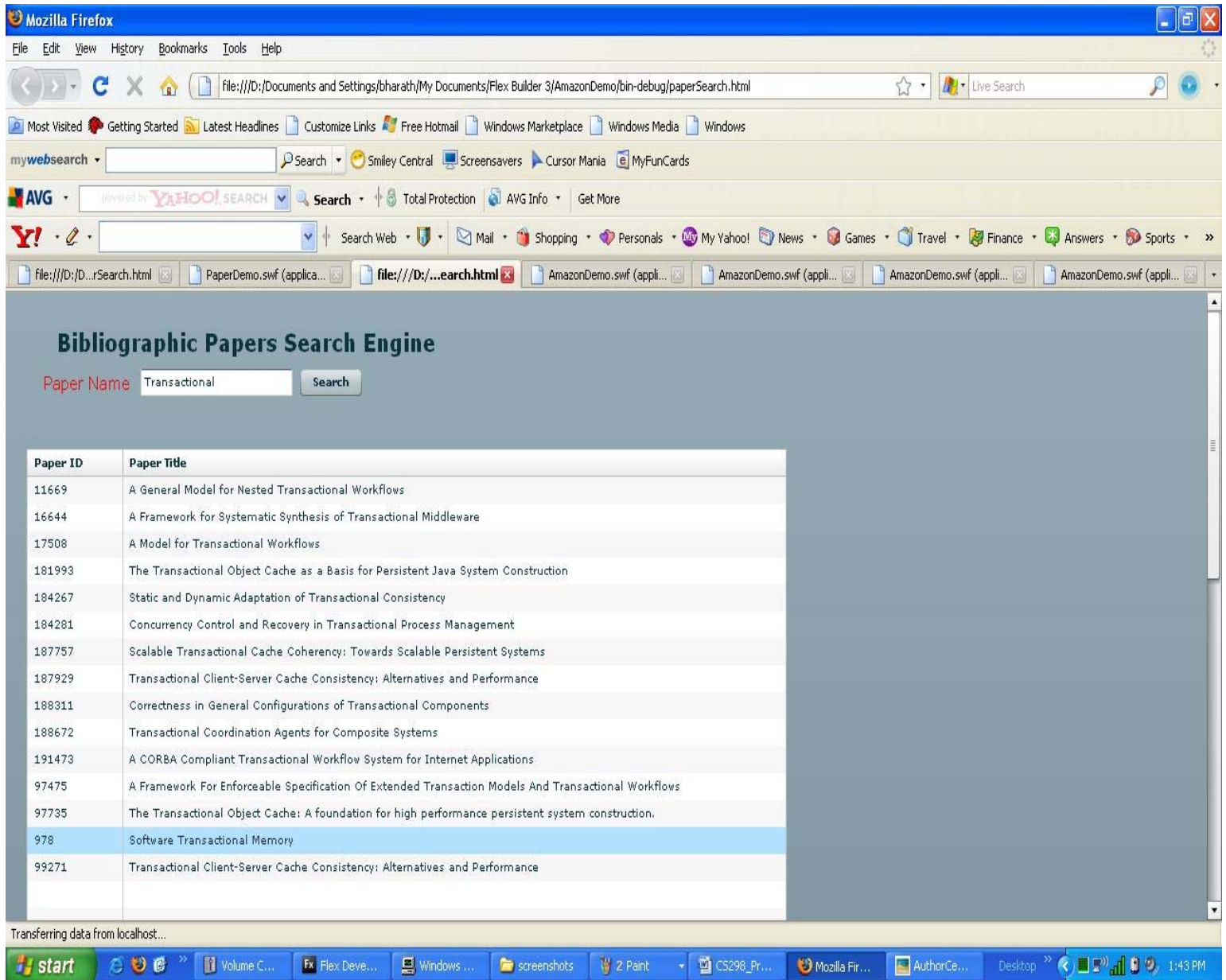
PAPER SEARCH QUERY SNAPSHOT

Figure 10.1 Paper Query Search Results Snapshot
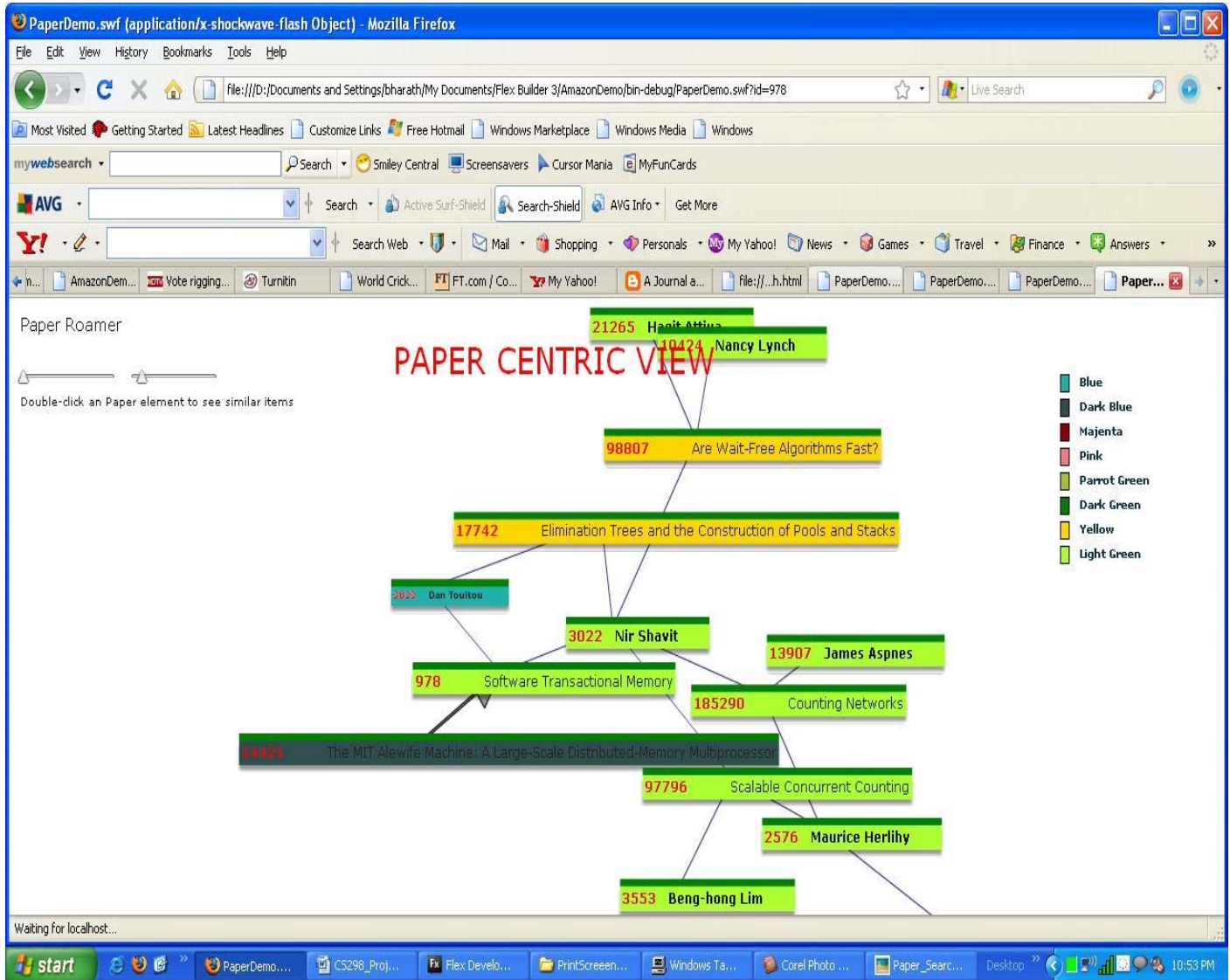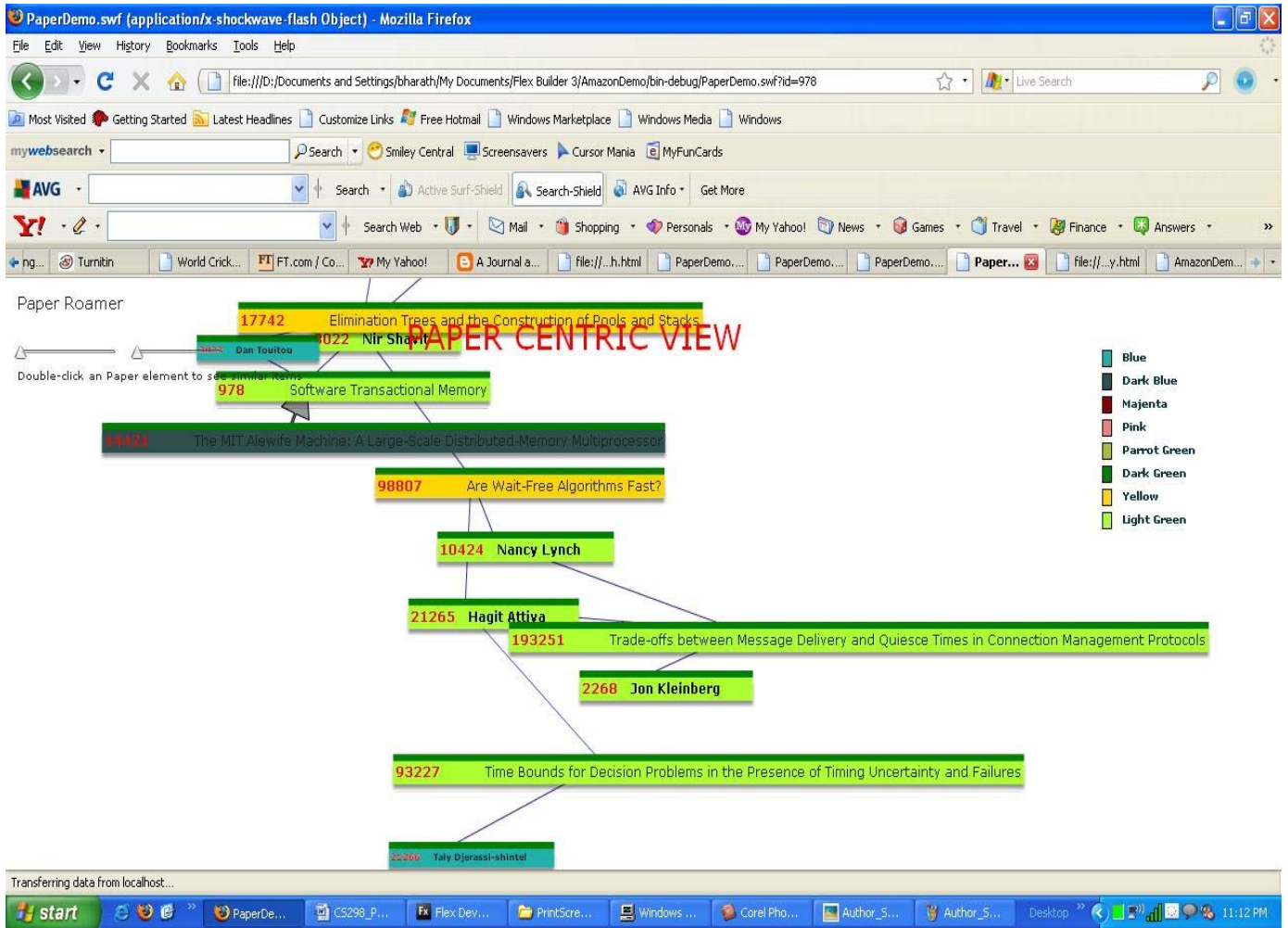
Paper- Centric View Snapshot

Figure 10.2 Paper Centric View Snapshot – Main Focus – Paper ID -978

Paper Title - "Software Transactional Memory"

Figure 10.3 Paper Centric View of user clicked paper with new focus – Paper ID – 98807

Paper Title – "Are Wait-Free Algorithms Fast?"

User/Author Search Query Snapshot

Figure 10.4 Author Query Search Results Snapshot

Author Centric View Snapshot

Figure 10.5 Author Centric View Snapshot with user clicked Author focus as Author ID – 2576 Author Name – "Maurice Herlihy"

11.   <u>MAJOR GOALS AND DELIVERABLES</u>

➢ A search engine was setup for the user to query a keyword of a paper/author and the result of the search is displayed on the webpage.

➢ Paper-Centric Visualization Graphs are provided for a particular paper with multi-level hierarchy provided for the graphs to access its referenced papers by an arrow and authors of the papers and also the other papers written by those authors.

➢ The nodes of the graph are color-coded based on the popularity of the paper.

➢ Author-Centric Visualization Graphs are provided for a particular author along with its co-authors and also the papers written by these authors with multi-level hierarchy provided for the graphs.

➢ The nodes of the graph are color-coded based on the popularity of the author.

➢ Was able to provide a very rich Visualization for the papers and authors, so that the end-user can easily establish relationship between authors and papers.

➢ Also was able to apply the Visualization algorithm for placement of the graph nodes and get a nice view of the tree-like hierarchy of the author and paper.

➢ Finally integrating all together, an online visualization of the bibliographic records was created by the help of Flex as the Visualization Tool.

12. CHALLENGES FACED DURING THIS PROJECT

- ➢ Integration of Spring Graph component into Flex.

- ➢ Parsing of XML data by using regular expressions in flex needed a lot of effort.

- ➢ Making the Graph interactive to get multi-level tree structure.

- ➢ Drawing arrows in the layout by using vector functions, which was little complicated.

- ➢ Varying the repulsion parameters such that the force applied on the nodes by using the Force Directed Layout algorithm, should not make the nodes move far away from the layout.

- ➢ Preventing overlapping of nodes on the screen.

- ➢ Drag and scroll features were little difficult to implement.

- ➢ Had to deal with lots of queries to the SQL database [16], which used to slow down my system.

- ➢ Learning flex took a bit of time, as it's a new language with lot of features in it.

- ➢ Overall working on a big data mining project like this is real challenge as we deal with real-time data and we have to load them, extract sensible data and visualize them, which in itself is a big task.

13. FUTURE WORK / ENHANCEMENTS

➢ Improve the Force Directed Layout algorithm to completely prevent overlapping of nodes on the screen.

➢ Reduce the number of queries to the database to improve efficiency.

➢ Use a separate database server, which reduces the overhead and efficiency.

➢ Improve the quality of the graph nodes in the application.

➢ Make efficient use of the layout and draw more nodes and edges in the layout.

➢ Provide support for more than 500 nodes in the layout at once.\

➢ Represent keywords in the paper-centric view.

➢ Improve the search engine interface by improving the keyword search criteria.

14. CONCLUSION

This project used the concepts of both data mining and data visualization and helped in representing the related data got from data mining techniques to be represented in the form of graph structure by using the data visualization tool like flex. The challenging part in this project was to represent the related data in the graph structure on the web browser, which was a very significant achievement.

The main key progress done in this project was representing data in the web browser in the form of graph which would allow the user to look into lot of related/ linked data in a single screen rather than having to look into different web pages for similar information. Also the color encoding for the author nodes and paper nodes in the view was very useful for the user to know the importance of that particular author/paper. So overall it was a significant project with lots of emphasis on providing a very good prototype for use in the next phase of this project.

Data Mining and Data visualization are two concepts that have been in use since a long time and are still being used in various areas ranging from business to academics. A lot of research and thus lot of development has been made in both the fields over the years. While both the concepts can stand by themselves, they are more advantageous when used together and that is the essence of this project.

15. <u>APPENDIX</u>

AddItem onto the Spring Graph function – Act like Controller:

```
public function addItem(id: String, title: String, weight: String, linkTo: PaperItem): PaperItem {
var authorNames: String = "";
var newItem: PaperItem = new PaperItem(id, title, weight, authorNames);

paper.add(newItem);
if(linkTo != null) {
var data: Object = {settings: {alpha: 1.0, color: 8, thickness: 2, width: 1.0, height: 1}};
paper.link(newItem, linkTo,data);
}
s.dataProvider = paper;
s.autoFitTick();
return newItem;
}
```

View Part of the Code where the Nodes are Drawn:

```
<mx:HBox id="vBox" dropShadowEnabled="true"
        borderStyle="solid"
        borderThickness="0"
        backgroundColor="{calcColor(data.weight)}"
        width="100%"
        height="100%">
```

```
<mx:Label text="{data.id}" color = "red" fontSize="{calcFontSize(data.weight)}"
fontWeight="bold"/>
<mx:Label text="{data.name}" color = "black" fontSize="{calcFontSize(data.weight)}"
fontWeight="bold" />
<mx:Label text="{data.title}" toolTip="{data.id}" textAlign="center" styleName="text"
fontSize="{calcFontSize(data.weight)}"/>


</mx:HBox>
```

Color calculation based on weight parameter:

```
public function calcColor(weight: String): int {
if (parseInt(weight)==0)
  return 0xADFF2F;
        else if(parseInt(weight)<10)
             return 0xFFD700;
        else if (parseInt(weight)>10 && parseInt(weight)<15 )
              return 0x008000;
        else if (parseInt(weight)>15 && parseInt(weight)<20 )
            return 0xADBF2F;
        else if (parseInt(weight)>20 && parseInt(weight)<25 )
            return 0xF08090;
        else if (parseInt(weight)>25 && parseInt(weight)<30 )
            return 0x8B0000;
        else if (parseInt(weight)>30 && parseInt(weight)<33 )
            return 0x2F4F4F;
        else if (parseInt(weight)>34 && parseInt(weight)<40 )
            return 0x20B2AA;
        else
            return 0x20B2AA;

                }
```

Web Service Call:

```
var xmlPaper:XML = new XML();
var loader:URLLoader = new URLLoader();
var urlString: String =
"http://localhost/bibliows/BiblioWS.asmx/PaperCentricView?paperId=" +id;
var request:URLRequest = new URLRequest(urlString);
loader.addEventListener(Event.COMPLETE, client.getPaperInfoResult);
loader.load(request);
```

Model class which parses the data and then stores the data onto binding variables:

```
var app: PaperCentric = PaperCentric(Application.application);
var repString: String = event.target.data;


var newrepString: String = repString.replace('
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\" xmlns=\"http://tempuri.org/\" ','');
var myPatternlt:RegExp = /&lt;/gi;
var strReplacelt: String = newrepString.replace(myPatternlt,"<");
var myPatterngt:RegExp = /&gt;/gi;
var strReplacegt: String = strReplacelt.replace(myPatterngt,">");
var xmlPaper:XML = new XML();
event.target.data = strReplacegt;
xmlPaper = new XML(event.target.data);
this.title = xmlPaper.paper.title;
this.weight = xmlPaper.paper.weight;
authors = xmlPaper.paper.authors.author;
var authorNames: String= new String;
for(var i: int = 0; i < authors.length(); i++) {
        var author: XML = authors[i];
```

```
        var name: String = author.name;

        var addString: String = name + ",";

        authorNames+=addString;

                    }

this.authorNames = "\n" + "Authors: " + authorNames;

linkpapers = xmlPaper.paper.references.paper;
```

16. REFERENCES

[1] Google Scholar

http://scholar.google.com/

[2] Citeseer Link

http://citeseer.ist.psu.edu/

[3] Amazon

http://www.amazon.com/

[4] Twitter

http://twitter.com/

[5] Force Directed Layout

http://en.wikipedia.org/wiki/Force-based_algorithms

[6] Hierarchical Visualization Algorithm

http://www.graphviz.org/Documentation/NW01.pdf

[7] Twitter

http://blog.danmcweeney.com/29

[8] Flex

http://www.adobe.com/products/flex/

[9] Flex Intro

http://www.devarticles.com/c/a/Flash/Introduction-to-Flex/

[10] Macromedia Flash

http://www.adobe.com/products/flashplayer/

[11] Flex Architecture

http://ria.dzone.com/news/sketch-adobe-flex-architecture-capabilities

[12] Spring Graph

http://mark-shepherd.com/blog/springgraph-flex-component/

[13] Amazon Demo

http://mark-shepherd.com/SpringGraph/AmazonDemo/bin/AmazonDemo.html

[14] Graph drawing Algorithms

http://en.wikipedia.org/wiki/Graph_drawing

[15] Force directed layout

www.ecs.umass.edu/ece/labs/vlsicad/ece665/presentations/Force-Directed-Adel.ppt

[16] SQL server 2008

http://www.microsoft.com/sqlserver/2008/en/us/default.aspx