Vision Plan

VISUALIZATION IN SOCIAL NETWORKS AND LINK ANALYSIS USING A* SEARCH: AN EXPERIMENTAL APPLICATION

Version 1.1

Submitted in partial fulfillment of the requirements for the degree
MASTER OF SOFTWARE ENGINEERING

JINCHENG GAO

CIS895 - MSE Project
Department of Computing and Information Sciences
Kansas State University
## Change Log

<table>
<thead>
<tr>
<th>Version #</th>
<th>Changed By</th>
<th>Released Date</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 1.0</td>
<td>Jincheng Gao</td>
<td>November 6, 2008</td>
<td>First release</td>
</tr>
<tr>
<td>Version 1.1</td>
<td>Jincheng Gao</td>
<td>November 24, 2008</td>
<td>Modified the rationale of A* searching algorithm and goals</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Motivation

The purpose of this project is to analyze datasets associated with large graphs for applications in link analysis. This project will focus on analyzing the structural features of the graph for a graph layout task. Applications of this feature analysis and graph layout system include link existence prediction in social networks. The Laboratory for Knowledge Discovery in Databases has designed a web crawler to collect information of social networks from web sites. A social network is a graphical model whose nodes correspond to entities within a social structure and whose links correspond to interrelationships between them. The main objectives of this project are to display the relational information in a social network and implement search algorithms to find an optimal solution for the relationships. Finally, I will implement experimental modules that implement the evaluation requirements for the link analysis system. Specifically, they will measure the accuracy, memory usage and running time of the algorithm on test data.

1.2 Terms and Definitions

Social Network – a graphical model representing a social structure, made up of nodes that are connected by specific types of shared social entities, such as values, visions, ideas, and friends [11].

Web Crawler – a program or automated script that browses the World Wide Web in a methodical and automated manner. It is also known as a web spider, web robot, or web scutter [2].

Knowledge Discovery in Database (KDD) – also called Knowledge Discovery and Data mining; the process of automatically searching large volumes of data for patterns and associations using tools such as machine learning, classification, and clustering.

Laboratory for Knowledge Discovery in Databases (KDD Lab) - A group headed by Dr. William Hsu whose primary focus is data mining.

Sequence Diagram – A graphical design used to display the order in which objects interact during a certain period.

Unified Modeling Language (UML) – A standard notation used to describe real-world objects.
Use Case Diagram – A behavioral diagram defined by UML. It provides a graphical depiction of system functionality in terms of actors.

High-Dimension Embedding Layout – a graph is first embedded in a high dimensional space and then projected back to 2-D or 3-D space using principal component analysis (PCA) [7].

Spring Embedding Layout – a spring force is assigned to each pair of nodes. A repelling force is added between two closed nodes, and an attractive force is active between two nodes too far apart. The layout of the graph vertices is calculated by minimizing the energy function [7].

Radial Layout – a focus node is put at the center of display and the other nodes are arranged on concentric rings around the focus node. Each node lies on the ring corresponding to its shortest network distance from the focus [10].

2. Project Overview

2.1 Backgrounds

Social network services focus on building online communities of people who can share their interests and activities through the Internet [11]. For example, Denga’s LiveJournal, Google’s Orkut, and Facebook allow users to list interests and links to friends, sometimes annotating these links by designing trust levels or qualitative ratings for selected friends.

The KDD group led by Dr. Hsu has created an HTTP-based spider called LJcrawler to harvest user information from LiveJournal to analyze the social network features. LiveJournal is one of the most popular weblog services, with a highly customizable and flexible personal publishing tool used by several million users [3]. A multithreaded version of this program, which retrieves BML data published by Denga, collects an average of up 15 records per second, traversing the social network depth-first and archiving the results in a master index file [4]. Understanding a network’s structure and analyzing its features poses a challenge in the case of large social networks containing tens of thousands to hundreds of millions of nodes.

One common approach to graph visualization is to use 3D representations or distortion techniques [6] to fit a large number of nodes in a single view. Most of these approaches, such as the Hyperbolic Browser [5] and the Core Tree [8] require a tree structure with fixed parent-child relationships. An alternative to fitting an entire graph into one view is to provide interactive exploration of subregions of the graph. Yee et al. [10] introduced a
new approach for animating the transitions from one view to the next in a smooth and aesthetically appealing manner for dynamic visualization of social networks with radial layout. However, the radial layout needs a polar coordinate system.

Feature analysis provides a basis for understanding a social network. The links provide social connections between users or communities in a social network. The best all-pair shortest path (APSP) search algorithm is Johnson’s algorithm, which achieves worst-case asymptotic running time $O(V^2 \lg V + VE)$ which is asymptotically faster than the $\Theta(V^3)$ a sparse graph, where $E \in O(V \lg V)$ [2]. The computing complexity is large for all pair search algorithm even in a middle size with 10-50K nodes. Actually, a social network is not all-pair edge connection, the number of candidate edges is nearly constant ($k=20$), which is much smaller than the number of nodes in a social network [4]. Thus the complexity of Johnson’s algorithm becomes $O(k(V \lg V + E))$ or $O(V \lg V + E)$. A* search algorithm can improve the computing complexity into $O(V+E)$.

A* search algorithm has been implemented by the KDD Lab to find the optimal solution from social network services. A* is complete, optimal, and optimally efficient among all algorithms using a given admissible heuristic [8]. No other algorithm expands fewer nodes than A* with the same heuristic function. It also expands nodes only once. The complexity of A* is $O(k(V+E))$ or $O(V+E)$ in the worst case. A* is better than Johnson’s search algorithm for a sparse graph ($E \in O(V \lg V)$) with a large number of nodes. However, the computer time and memory cost are the drawbacks of A* algorithm. As it keeps all generated nodes in memory, A* usually runs out of space long before it runs out of time [9]. Therefore, the A* algorithm need to be tested with a large number of nodes. The main purpose of this project is to analyze the A* search algorithm with a particular admissible heuristic and different sizes of nodes. The experimental application of my system will measure the accuracy and running time of A*, and use it to visualize the network structure.

2.2 Goals

Three goals will be achieved from this project:

- 2D layout with spring embedding and high-dimension embedding approaches will be implemented to visualize large networks. As the A* search algorithm must be given the coordinates of nodes to compute their straight distance, radial layout
with a polar coordinate system is difficult to use. Thus, only spring and high-dimension embedding systems will be implemented for the layout.

- The shortest path will be highlighted after finishing a search procedure and the path will be saved in a graphic format.
- The search accuracy and running time of A* will be compared with that of the BFS and greedy search algorithms. I will test A* with 500, 1000, and over 2000 nodes of networks. My test data sets will consist of geographic, social, and biological networks.

2.3 Project Diagram

Figure 1 is the block diagram of this project. Users will use the GUI to interact with the search and visualization tools which are implemented in the Application Layer. The search and display services extract data from the Database in the Storage Layer. All of the tasks are performed on the Java Virtual Machine which is running on the user’s hardware systems.
Figure 1: Project Block Diagram
Figure 2 provides the general data flow of the program. Graph drawing and search processes will use vertex and edge data extracted from the database and position information from the layout algorithm to generate objects for visualization.
3. Project Requirements

The project requirements section will include all requirements of the project. Each requirement will be discussed in detail with an associated requirement number.

The requirement of this project will be classified into four categories: applications requirement, visualization requirement, search requirement, and testing requirement (Figure 3).

![System Use Case Diagram](image)

**Figure 3: System Use Case**

3.1 Application Requirements (API)

This section details all of the requirements related to the main applications of the project.

3.1.1 API100 [Critical Requirement]

This program will provide a graph user interface (GUI) for user interaction between systems and users.
3.1.2 API101

The application should have a menu bar that contains at a minimum: a File menu, processing, and a Help menu.

3.1.3 API102

The application process menu should have minimum functions: parsing a graph, search path, and reordering network structure.

3.1.4 API103

A* search running time function to detect the time used with different sizes of nodes.

3.1.5 API104

Memory cost detection function to detect the usage of memory

3.2 Visualization Requirements (VR)

3.2.1 VR201 [Critical Requirement]

Radial Layout is required to display the social network in a single view. An approach described in [6] will be implemented.

3.2.2 VR202 (optional)

Spring Embedding Layout will be implemented in the interface. The spring embedding algorithm assigns force between each pair of nodes. When two nodes are too close together, a repelling force comes into effect. When two nodes are too far away, an attractive force is generated.

3.2.3 VR203 (optional)

Multiple-Dimension Layout will be implemented in the visualizing function. A graph is embedded in high-dimensional space, then projected back to two or three-dimensional space.

3.3 Search Requirements (SR)

3.3.1 SR 301
The A* search algorithm will be implemented in the GUI-driven application.

3.3.2 SR302
Greedy best-first search is optionally added in the main interface.

3.4 Testing Requirements (TP)

3.4.1 TP401
A small network (with 500 or fewer nodes) is required to test the A* search algorithm.

3.4.2 TP402
A mid-sized network is required with between 500 and 1000 nodes.

3.4.3 TP403
A larger network (> 2000) is required to test the scalability of the layout algorithm and rendering system.

4. Assumptions

- Java Runtime Environment 1.3.1 or later will be installed on the computer running the application.
- In order to run a search, the user will have an active Internet connection.
- In order to perform a Web Crawl in a reasonable amount of time, the user will have a high-speed Internet connection (DSL or better).
- The user will need a minimum of 512 MB of memory.
- The user will have a computer with a minimum speed of 1.6 GHz.

5. Constraints

- Java will be used for the web crawling. While it will not be as efficient as using other languages, there is much web functionality defined in the JDK, making it easier to write the web crawling.
- Entity Search is limited to searching for contact info entities. An excellent future enhancement would be to add other entity types.

6. Environment

- Eclipse 3.3.0 will be used as the IDE.
- Java version JDK 1.5 will be used.
7. References


