

CSC 445: Spring 2008: Projects

Introduction

The research projects are for the honors students and for those who would like to try their hand at “research”. Small groups of 2-3 students will work on them. The projects typically involve designing, implementing, and testing an algorithm. Should the project lead to a publication, it will increase your credentials and possibly lead to a conference trip (in the last few years, students working on projects have gone to San Diego, Canada, Switzerland, Italy, Portugal, Australia). I will expect a project plan, a progress report, and final paper:

1. a 2-page project plan (due February 11)
2. a 5-page project progress report (due March 24)
3. a 10-page final paper (due May 5th)

Each of these projects should lead to a conference/journal publication and the final report is a close-to-final draft of this publication. All three of these reports must be written in LaTeX and must include Figures (illustrating claims, or proofs, or examples), Images (illustrating screen-shots, or algorithm output), and Tables (illustrating performance, or results).

NOTE: The project grade is worth 20% of your grade for this class!

1 Circle and Spherical Layouts

Some of the most flexible algorithms for calculating layouts of simple undirected graphs belong to a class known as force-directed algorithms. Also known as spring embedders, such algorithms calculate the layout of a graph using only information contained within the structure of the graph itself, rather than relying on domain-specific knowledge. Graphs drawn with these algorithms tend to be aesthetically pleasing, exhibit symmetries, and tend to produce crossing-free layouts for planar graphs. However, existing force-directed algorithms are restricted to calculating a graph layout in Euclidean geometry, typically R^2 , R^3 , and more recently R^n for larger values of n . There are, however, cases where Euclidean geometry may not be the best option: certain graphs may be known to have a structure which would be best realized in a different geometry, such as on the surface of a sphere or on a torus. Furthermore, it has also been noted that certain non-Euclidean geometries, especially hyperbolic geometry, have properties which are particularly well suited to the layout and visualization of large classes of graphs. In an earlier paper [1] we presented a method by which a force-directed algorithm can be generalized so that it can compute a graph layout in any of a large class of geometries (known as Riemannian geometries), so long as the mathematics describing how the geometries behave are well described. Our method relied on extending the Euclidean notions of distances and angles to Riemannian geometries via projections to and from appropriately chosen tangent spaces. Both the hyperbolic and spherical cases were implemented in the GraphAEL system.

This project will focus on obtaining circular and spherical layouts as a plug-in to the yFiles visualization system. The main problems will be obtain “center-free” layouts. The “central” part of a drawing naturally leads viewers of the drawing to attach greater importance to vertices and labels placed in the center. In many naturally occurring graphs there is no such center, and drawing the graph in the conventional way can be misleading. Circular layouts (where all the nodes are placed on the circumference of one or more circles) and spherical layouts (where all the nodes are placed on the surface of a sphere) are naturally centerless.

2 Using Graphael to visualize dynamic graphs

Graphael is graph drawing system designed to provide the necessary structure and flexibility for force-directed graph drawing research. The system provides a set of core algorithms and visualization routines, as well as a powerful interface that allows the user to combine different algorithms and visualization methods, and to easily add new ones. In addition, the system contains several novel algorithms and visualization techniques, such as force-directed methods

in hyperbolic and spherical spaces, and techniques for dealing with graphs that evolve through time. The system is written in Java and is available as a downloadable program or as an applet at <http://grapael.cs.arizona.edu>.

While static graphs arise in many applications, dynamic processes give rise to graphs that evolve through time. Such dynamic processes can be found in software engineering, internet/telecommunications traffic, and social networks, among others. Although we have been successful in using the force-directed method for visualization of evolving graphs [2], our approach quickly runs into scalability issues, as the number of timeslices grows large. We address possible ways to deal with these issues. We also consider extensions of our approach to dynamic graphs.

When visualizing evolving graphs we have all the data in advance. We would like to use the knowledge of the changes that the graph would undergo to pre-determine the positions of the vertices at any given moment in time. Therefore, our approach for visualization of evolving graphs relies on the idea of creating a merged graph from the series of input graphs. In many applications both the graphs are large (tens of thousands of vertices and edges) and they are made of many timeslices (thousands or tens of thousands). For example, in visualizing the behavior of dynamically modifiable code, individual graphs have thousands of vertices and tens of thousands of timeslices. Although our force-directed algorithms work well for large graphs, they do not scale to graphs with millions of vertices and edges.

When dealing with a large number of timeslices, the influence of the first few on the last few is minimal. Thus, it is not necessary to take all of them into account when computing the layouts. A natural way to deal with a large number of timeslices would be to break them into groups of constant size, and compute an average layout for each group. If the number of groups is still large, we can repeat the process, until we are left with a small number of groups. This approach can give us a sliding window of influence for each individual graph, and would allow us to extend our approach to much larger instances of evolving graphs.

In addition to allowing us to deal with very large evolving graphs, the above approach can also help us in visualizing dynamic graphs: Recall that evolving graphs correspond to the off-line version of the problem, where we know each graph in the series in advance. This allows us to preprocess the data and precompute the layouts of each graph. In contrast, dynamic graphs correspond to the on-line version of the problem, where graphs in the series are generated in real-time. In this case we cannot precompute the layouts, however, the two criteria of readability and mental map preservation can still be used to measure the quality of the visualization. One possibility of extending the approach for evolving graphs to dynamic graphs is to build the merged graph on the fly. Each new graph in the series would be integrated in the merged graph and a new layout would be computed for the entire merged graph. Vertices would be initialized to their positions from the previous layout and the force-directed method would be applied. As a result, the layout of each of the graphs in the series would change and in particular, we would obtain a layout for the current graph.

3 Social Networks Data Analysis

A great deal of social networks data is available: Facebook, Myspace, Orkut, LinkedIn, and other social networking sites are good examples. Many data-sets of this type are large and dynamic – they evolve over time. This project will focus on collecting data from two such sources, studying and comparing the properties of the corresponding networks, and visualizing the data. Collecting the data can be done by writing a simple web-spider or by utilizing existing ones. Starting with basic properties (such as mean and median vertex degree, clustering coefficients, diameter) we will also look for new metrics and study the similarity and differences in the two sources. The visualization component will be built on one our existing graph visualization platforms but will also add features such as layouts based on graph cores, different centralities, etc. Control over the size of the displayed graph will also be incorporated (e.g., dropping low degree nodes, clustering distant groups, and so on). In an earlier paper we studied the collaboration graph of graph drawing researchers [3]; this is a good starting point for the kind of work we will be doing in this project.

Bibliography

1. G. Kobourov and K. Wampler "Non-Euclidean Spring Embedders," 10th Annual IEEE Symposium on Information Visualization (InfoVis), p. 207–214, 2004.
2. D. Forrester, S. G. Kobourov, A. Navabi, K. Wampler, G. Yee, "grapael: A System for Generalized Force-Directed Layouts," 12th Symposium on Graph Drawing (GD), p. 454-466, 2004.
3. C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee, "GraphAEL: Graph Animations with Evolving Layouts", 11th Symposium on Graph Drawing (GD), p. 98-110, 2003.