User-Guided Interactive Graph Layout

Julia Ferraioli

August 11, 2005

1 Introduction

Current graph visualization techniques gather information found in a database or a file and create output on the screen that represents that data. There are different types of layouts from which to choose, such as force-directed layouts, circle-layouts and spring layouts. These layouts are all either random, or based upon the edges in the graphs. They do not lend any weight to what the nodes might represent. By making the graph interactive, thereby allowing the user to manipulate the graph, the graph slowly develops into the organization which the user intends. However, with this method, the user would have to rearrange every node in order to accomplish the intended arrangement. This is inefficient and frustrating to the user.

If we incorporate the attributes into the arrangement of the nodes, we should be able to arrive at the correct arrangement much faster than without taking the attributes into account. This may be accomplished by various means, but the two that we considered for this research are simple clustering and constrained clustering. Just performing simple clustering, by using an algorithm such as k-Means, does allow the graph to take attributes into consideration, but does not allow user feedback. By employing an algorithm such as COP-KMeans or PCKMeans, we take both attributes and user feedback and use them to arrange the on-screen graph. We propose that combining a user-friendly interface with some type of constrained clustering will allow the user to arrive at the desired graph in a significantly shorter amount of time.

2 Existing Clustering Techniques

There were two primary clustering algorithms that we considered when conducting our research. The first algorithm is COP-KMeans, developed by Wagstaff et al. [3], which takes a matrix of "must-link" constraints and "cannot-link" constraints. Each pair of nodes has a "must-link" constraint, a "cannot-link" constraint or no constraint at all. The algorithm then clusters using k-Means, while ensuring that none of the cluster assignments violate the constraints. If the constraint matrix is such that the graph is over-constrained, the algorithm fails. The second algorithm is PCKMeans, developed by Basu et al. [1]. PCKMeans

also takes a constraint matrix of "must-link" and "cannot-link" constraints, but the constraints can carry weights. This algorithm *can* violate constraints, but it imposes a penalty upon itself for doing so. It clusters using k-Means while minimizing the incurred penalty.

These two clustering algorithms fall under the category of "semi-supervised clustering" since in both cases there is a definite goal in mind. These algorithms are semi-supervised, in the sense that they use constraints to make decisions, not that they have any real human supervision.

Cohn et al. address the issue of incorporating user feedback [2] when clustering data, which does have an actual person overseeing the organization of the data. Their approach allows the user to define the metric in which to cluster. While we do not use their method of user-defined metrics, our goals coincide in providing a way for the user to interact with the data.

3 Aims of the Research

The primary aim of this research is to allow the efficient organization of information while considering the goals of the user. This ability can help users search through databases in order to find the information that they are looking for in a short period of time, it can organize databases in an efficient manner, and it can make judgments on how to organize based on more than one definite attribute. We intend to incorporate actual human supervision with semi-supervision to accomplish our aim.

A secondary aim of our research is to explore the ways that a graph may be visualized, and how this may impact a real human user. For example, if the nodes are colored based on characteristics, does that assist or impede the user? We wanted to analyze what sort of elements, or what combination of elements, would expedite the arrival at the user's idea of the correct graph. Could a combination of shapes, colors and images work for the user?

However, our primary aim consumed most of our time, and the possibility of user testing quickly became an impossibility within the time limit placed upon us. Fully exploring the graphical element was also time-consuming, and we decided to concentrate on our primary aim.

4 Packages and Layout Implementation

In our program, we use the package Prefuse to handle the visualization of the nodes, and the package WekaUT to handle the clustering aspect and to evaluate the results. Prefuse renders the images, the animation and the interactive aspect of the nodes. The user has the ability to drag a node to a specified location on the screen and "lock" it in place. It also has other features such as different types of layouts, the ability to "highlight" nodes and edges in response to user input and to read in files through different formats. WekaUT includes the standard Weka package, and adds to it different clustering algorithms such as k-Means,

PCKMeans, MPCKMeans and XMeans.

Our layout is a force-directed, spring-embedded layout, which visualizes the nodes with a simple label of the node identification. The data files which we use are standard XML files using the convention for formatting graphs. Each attribute name in the XML file is preceded by a symbol indicating the type of attribute value it represents. Categorical data is preceded by "@" and numerical data is preceded by "#." Data that is to be ignored is preceded by "$." The program automatically ignores the ID and label of the node, as they are not used for clustering. Prefuse takes the data provided in the XML file and creates a visualization based on it.

5 Metrics

The metrics implemented in our program follow two basic cases.

Case 1: Data to be averaged is numerical.

Case 2: Data to be averaged is categorical.

If the data is numerical, then we use the standard Euclidean Distance metric which is $\sqrt[2]{\sum (x - x^1)}$. If the data is categorical, then we use a distance metric that returns a distance of 0 if the data being compared is the same, or 1 if the data is different. In many data sets, numerical and categorical data are mixed. If they are, then we simply differentiate between the two types of data when calculating distance.

The program does not take values other than numerical or categorical into account. Therefore, it will not calculate the difference between a set of strings to tell how far apart each of them are from each other. This type of data would be preceded by "$" and disregarded.

6 Constraint and Cluster Generation

We gather the data about the nodes, normalize it, and create an Instance for each node. We then associate each Instance with an Instances. After this, the program captures the x, y coordinates of each node on the screen. When the user moves a node, it calls several methods including itemPressed, itemDragged and itemReleased. If only one node is moved, the program continues as before. If more than one node is moved, itemReleased generates constraints, calls PCKMeans and creates nodes for the cluster centers and edges to those in the appropriate cluster.

The constraints in our program are dependent on two variables, which are δ and ϵ. When two or more nodes are moved, the most recently moved node is compared to all of the other nodes which have already been moved. If the distance between the compared nodes is greater than δ, then a "cannot-link" constraint is generated for that pair. If the distance between the compared nodes is less than ϵ, then a "must-link" constraint is generated for that pair of nodes. If the distance between the two nodes is less than δ but greater than

ϵ, then no constraint is generated. Essentially, the user tells the program that s/he definitely does or does not want these two nodes together.

When itemReleased calls PCKMeans, it feeds the current constraints into the algorithm. If only two nodes have been moved, it passes only one constraint, provided that the distance between the two nodes is greater than δ or less than ϵ. PCKMeans returns with cluster assignments, which we then use to generate nodes and edges. A cluster center is represented by a node with the label "cluster" and a greater mass than the other nodes. Its attribute space coordinates are the mean of the numerical attributes and the mode of the categorical attributes of the nodes in its cluster. The on-screen coordinates of the cluster centers are the mean of the nodes connected to it that are fixed. If only one node in the cluster is fixed, then the cluster center is directly on top of that node. The edges start from the cluster center and go to the nodes that are in that cluster. The edges have a weight of "3" in order to make their spring settings stronger.

Each time an item is pressed, the itemPressed checks to see if there are any cluster centers in the graph. If there are, those nodes are destroyed, along with their edges. This way, every time that PCKMeans is called, it adds nodes to the original graph. By running PCKMeans every time the user clicks on a node, the algorithm is constantly updating the cluster assignments based on a growing set of constraints.

7 Visualization

For the experiments, we leave the cluster centers and their edges visible, so that we may also observe the process rather than only analyzing the results. In the final project, the cluster centers and their edges are set to the same color as the background, effectively making them invisible to the user. In the future, we could give the items in the same cluster some sort of visual attribute to identify them specifically.

8 Experiments

We have designed four sets of experiments to test the effectiveness of this approach.

1. A purely manual sort. This experiment does not use a force-directed layout, it does not cluster, it picks a node at random and places it near the other nodes that are of the same class (cluster) type.

2. Uses the force-directed layout, but it does not cluster. It selects a node by choosing the one that is farthest from the other nodes that are of the same class.

3. Uses the force-directed layout and does cluster, but it chooses nodes at random.

4. Uses the force-directed layout, clustering, and the farthest-first heuristic for choosing nodes to move.

To evaluate the performance of these experiments, we are using the Adjusted Rand Index (ARI) to measure the accuracy of cluster assignments. The ARI returns a value between -1 and 1 each time. A return value of -1 means that the clustering assignments were completely wrong, and a return value of 1 means that the clustering assignments were exactly correct. The ARI is called in the first two experiments every time a node is moved, or in the second two, every time PCKMeans is called.

Our experiments essentially measure how quickly the ARI approaches 1. Every experiment will achieve an ARI value of 1 if all of the nodes are moved correctly. We assume that the user is going to put the nodes in the correct cluster, based on their understanding of the classification of data. Our expectations of these experiments is that their performance will rise in ascending order, the first having the lowest performance. Indeed, preliminary results support this supposition.

The Adjusted Rand Index for a 13 Node Data Set

This table shows the results of the four experiments over a five run period. The resulting ARIs for each experiment and each run are averaged together and entered below.

$NodesMoved$	$Experiment1$	$Experiment2$	$Experiment3$	$Experiment4$
1	0.06	-0.08	0.0	0.0
2	0.11	-0.03	0.52	0.54
3	0.12	0.06	0.55	0.57
4	0.18	0.13	0.57	0.61
5	0.27	0.28	0.64	0.61
6	0.33	0.39	0.64	0.74
7	0.4	0.54	0.73	0.82
8	0.47	0.69	0.83	0.88
9	0.56	0.81	0.87	1.0
10	0.7	0.91	0.96	1.0
11	0.78	0.96	0.96	1.0
12	0.91	0.96	1.0	1.0
13	1.0	1.0	1.0	1.0

These results support our expectations, but these are also the first set of experiments that we have run whatsoever. The data set used above is part of the famous iris data set, cut down to a manageable size and translated into an XML format.

We have not yet run any experiments such as above with categorical data, and since with a mix of categorical and numerical data we would be mixing distance metrics, it is difficult to predict the outcome of such an experiment. There is no reason to expect that the number of dimensions of the nodes should have any significant impact on the performance of the program.

9 Conclusion

The design phase of this project has ended, but the experimentation phase is just beginning. It would be hasty to judge the outcome of this project based on one data set and five experiment runs. We are still working on the final paper for this project, pending the outcome of more experiments with more datasets. We hope that we will finish in time to submit the paper to the 2006 International Conference on Intelligent User Interfaces, either as a short or a long paper. However, it seems logical that these preliminary results would reflect the results of other data sets of a similar type.

It seems as though the combination of visual supervision and semi-supervised clustering yields in general a better result than semi-supervised clustering alone, and then to combine this with a force-directed layout performs even better. If these conclusions hold true, this could have an impact on such services as search engines, since visualization of the web is becoming more popular and in demand.

Future work includes more experiments, and the integration of the design aspect with the organizational aspect. Finding an optimal balance between the two would make the program even more effective to the user. By working with people in computer graphics, this could turn into a product that would be seen implemented outside a laboratory setting.

References

[1] S. Basu, A. Banerjee, and R. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM International Conference on Data Mining*, pages 333–344, FL, USA, April 2004.

[2] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. Technical Report TR2003-1892, Cornell University, 2003.

[3] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In *Proc. of 18th Intl. Conf. on Machine Learning (ICML-2001)*, 2001.