College Football Games as a Network in NodeXL

For the network visualization assignment in CMSC 734: Information Visualization, a dataset of college football games spanning 10 seasons was examined. The dataset consisted of every college football game where at least one of the teams was in the Football Bowl Subdivision (FBS) from the 2000 to the 2010 season. The data being considered was only a small subset of college football data available [1]. Due to conference re-alignment, various NCAA recruiting scandals, and the flux of program strength limiting the scope of data was necessary for meaningful results. In total there were 221 teams playing a total of 7598 games. Many games were repeats due to the structure of the college football conferences and the practice of scheduling non-conference rivalry games. Only 1206 pairs of teams were unique in the 10 year period. Game instances were annotated with the date the game occurred, the home team, the away team and each team’s score. The data was provided in a white-space separated format that could be imported into NodeXL.

A graph was constructed where the college football teams were nodes and edges were individual instances of games. Depending on the questions being asked, additional information could be added to the graph. For example, to denote who won a game edges could be directed, pointing to the winning team. Edge weights could be added to denote the score difference. Close games would have low edge weights, and blow-outs would have larger edge weights.

Findings regarding the strengths of teams and structures of conferences can be verified by statistics and expert polls accessible on ESPN’s website [2].

Identifying BCS and FCS Teams Visually

The data consist mostly of games were Bowl Championship Series (BCS) play other BCS teams. However, for season openers and other non-conference a small number of games are played between BCS teams and Football Conference Subdivision (FCS) teams. It is always an upset if an FCS team beats a BCS team.

Due to the number of games a FCS team plays against FBS opponents, it was easy to identify these teams. The Fruchterman-Reingold layout scheme placed less central teams, with a lower degree further from the center of the graph than better connected teams. Fruchterman-Reingold was applied repetitively to get optimal placement. The vertexes were then colored red for FBS teams and blue for FCS teams to verify this finding visually. Additionally, when a directed graph was used the majority of edges point from the blue, FCS nodes, to the red, FBS nodes. This indicates that most of the times FBS teams beat FCS teams. Figure-1 shows the graph with the FCS and BCS teams colored accordingly.
Identifying Divisions within Conferences using Cliques

Every season college football teams play every other team in their conference. Using a simple undirected and unweighted edge representation, cliques that are larger than the minimal size of a conference would indicate a conference. With the exception of a few independent teams, teams do not have enough non-conference games to play every team in a different conference. False positives are not expected.

A process was developed to using an undirected graph representation to identify football conferences in the data. The process is as follows:

1) Select a single season to work with. Conferences have changed in this time period, so this step was added so that accurate conferences for a single season.

2) Group by clique motif with between 5 and 15 members.

This works really well to identify college football conferences and college football divisions within conferences. A 13-clique identified the Ohio Valley Conference. Similarly, a 12-clique identified all teams in Conference USA. Smaller cliques corresponded to divisions in the other conferences. Figure-2 shows a clique that consists entirely of the Atlantic Coast Conference (ACC) teams. Figure-3 shows a graph where the various cliques, corresponding to conferences are shown with X shaped colored figures.

An initial attempt to identify conferences involved graph clustering. The clustering produced 3 clusters that each consisted of multiple conferences. There were a sufficient number of games played between conferences so that multiple conferences satisfied the definition of a cluster. A clique is a more strict definition than a cluster, so cliques were better able to identify football conferences and divisions.

An example ACC clique is shown in Figure 2. A graph showing cliques and interactions between cliques is shown in Figure 3.

<table>
<thead>
<tr>
<th>Virginia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virginia Tech</td>
</tr>
<tr>
<td>North Carolina State</td>
</tr>
<tr>
<td>North Carolina</td>
</tr>
<tr>
<td>Miami (Florida)</td>
</tr>
<tr>
<td>Duke</td>
</tr>
<tr>
<td>Wake Forest</td>
</tr>
<tr>
<td>Clemson</td>
</tr>
</tbody>
</table>
Geography Does Not Matter When Playing Non-Conference Games

Traveling long distances to play non-conference games would be costly and time consuming. Do teams tend to play non-conference games against geographically distant opponents?

Conferences were identified using the clique technique described in a previous section. Conferences were displayed as large X’s motifs. The conferences were then laid out geographically by hand. Edges between X’s indicate non-conference games. The density of the graph in figure 4 shows that non-conference games tend not to be against geographically close teams.
Without a playoff system, ranking college football teams has been controversial and challenging. The Bowl Championship Series (BCS) uses a secret combination of human polls and statistical ranking to select teams for the national championship. These human rankings and statistical rankings never fully agree.

Authority ranking algorithms such as PageRank could be applied to determine the best football team for a given season. Page rank was run using a directed representation of the college football game graph, where edges went from losing teams to winning teams. PageRank was run with two different edge weights.

Results of the PageRank algorithm were not perfect. For example, Ball State was ranked in the top 5. Ball state had one good season where they only lost one game, but are in general not a very good team. Alabama, Syracuse, Florida State and Virginia Tech ranked well and had good seasons in the 2000-2010 decade. The rankings can be justified, but no expert would say that these rankings are good.

A follow up idea was to run page rank on a smaller set of teams, for example a conference. NodeXL did not support slicing the data and then re-running page rank on that subset of data. Completing these actions would require re-importing only the subset of data under consideration.

![Figure 5 - PageRank for Ranking Teams](image)

**Review of NodeXL as a tool**

The NodeXL network visualization tool was used to analyze this dataset. In general this tool was very easy to use, but was not very polished. The interface was reviewed with consideration for Shneiderman’s “Eight Golden Rules of Interface Design” [3]. Specific positives and negatives are noted in sub-sections, below.

- **NodeXL Positives**
  1) NodeXL had a low barrier to entry. It was very quick to import a dataset from a specific source and start analyzing data.
  2) Microsoft Excel is an accepted office tool. NodeXL had support most of Excel’s shortcuts satisfying the interface design goal, ‘Enable Frequent Users to Use Shortcuts’.

- **NodeXL Negatives**
  1) As I was updating the data in the worksheets, the graph did not update. A button had to be pressed to re-draw the graph. After making 3 or 4 parameter changes it was difficult to understand what affect each parameter had on the re-drawing of the graph. In general, NodeXL lacked informative feedback at each step.
  2) It was not clear how the algorithms were changed to suit different graph representations. For example, it was unclear if PageRank was considering directed graph edges and edge weights.
The help was insufficient. Searches for ‘PageRank’, ‘pagerank’, ‘Page Rank’ and ‘page rank’ did not return any results. The program lacked an internal locust of control, when I was using the PageRank features I did not feel like I was in control of the program.