# Rating Strength in Competitive Networks

### 1 Introduction

Teams engaged in various competitions are often ranked based on simple criteria such as win percentage. In diverse networks in which not all opponents play each other, this method can be deficient. Not only will it be difficult to separate out teams with equal win percentages (which in a short season can be relatively often), but in networks segmented into divisions with varying skill levels- the statistic can actually be misleading. What we need is a consistent measure of performance that makes the best possible use of all games throughout the network to rank each team. One of the most notable areas of impact is NCAA football. 117 teams are ranked at the end of each year, based on only 12 games per team in order to determine who will play in the national championship playoff of only 4 teams. In this context, it is important that any technique used to rank teams be free of bias as well as accurate.

This paper investigates the application of various techniques to rank the strength of sports teams or individuals engaged in competition. I want to approach the problem from the perspective of an organization trying to fairly rank the top teams in a league. Thus, all of these algorithms will be limited in scope to the following information

* Wins and Losses
* Home field advantage
* Margin of Victory
* Team momentum (i.e. placing priority on victories later in the season)

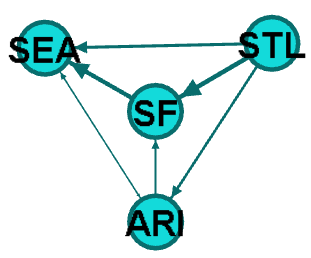
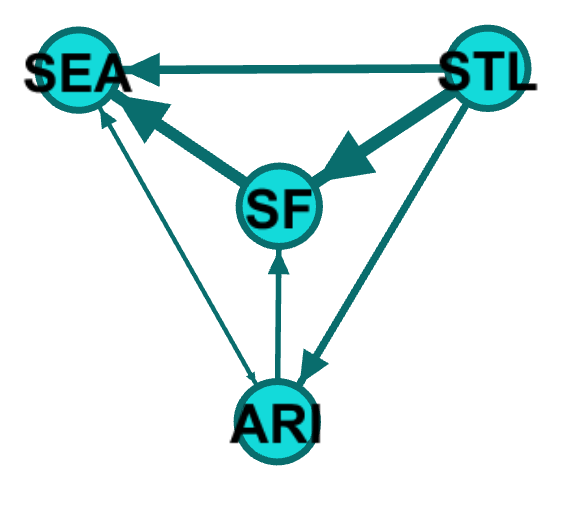
I’ll remain blind to factors such as historical team or conference strength, player injuries, or advanced statistics measuring efficiency at a play or player level.

I’ll introduce three separate data sets used for analysis and explore several techniques for rating the relative strength of teams. I’ll then judge the relative merit of each of these rating systems based on their ability to accurately predict the winner of an upcoming game or match in each of three different leagues. I’ll briefly explore how measuring certain aspects of a competitive network can inform inputs to my prediction methods- using simulated seasons for comparison.

## 2- Competitive Networks

First, I want to introduce the concept of formulating competitive networks into a directed graph. The graph has each team as a node, and each game as a directed edge pointing from the loser to the winner. If a team beat another team in two separate games, then one edge would be drawn between the teams with an edge weight of 2. See figure 1.1 example based on four NFL teams from the 2013 season (the NFC West composed of Arizona, Seattle, San Francisco, and St. Louis). Seattle beat St. Louis twice, and went 1-1 against both San Francisco and Arizona, and so on. I will used this formulation to take advantage of many properties of network graphs- particularly properties of the adjacency matrix- to make judgments about the strength of teams.

An extension of this technique could include the point margin of each game as an edge weight as in figure 1.2. There is some amount of information loss with this method because now if a team wins two games against the same opponent, then the edge weight is simply the sum of the point differentials. So while we see that there was a large total margin between St. Louis and San Francisco, we can’t tell if this was because of a single blowout or two combined victories. Of course point margin isn’t the only type of edge weight we could apply to represent strength of victory. Ideally, each edge should represent as best as possible measure of strength of victory over another team, accounting for any the systemic advantage given to either team (such as home field advantage).

Figure 1.1 Equally Weighted Games Figure 1.2 Win Margin Weighted Games

### 2 Rating Algorithms

The task at hand is to be able to predict the outcome of a game between any two teams in a competitive network. In other words, if we are to add an edge between any two nodes in our graph, we want to predict the direction in which it will point.

* 1. **Traditional Methods**

Win Percentage-

As stated this is the most straightforward mechanism for rating teams- simply the % of games in which a team has won.

Strength of Schedule (SOS)-

This method incorporates the winning percentage of

* 1. **Exponentiated Win Percentage**

Let’s examine the meaning of the adjacency matrix in the context of this problem. Row A in column B of the matrix is an edge between node A and node B- representing a victory for team A over team B. This is a helpful measure when predicting future games between A and B, but it likely has a small (if not 0) sample size.

If we square the adjacency matrix we can find the number of edges of length 2 between A and B. This represents the number of teams team A has beaten who have subsequently beaten team B. These too will be helpful data points but should be given less emphasis than more immediate games between A and B. Cubing the adjacency matrix will give us a similar measure with 3 degrees of separation, and so on. So let’s take a look at an infinite sum of the powers of the adjacency matrix, discounting each subsequent power, n by :

This makes use of a convenient convergence property, but may not include the right discount factor for our data. We can scale our adjacency matrix by an adjustment factor, , which will change the relative importance of games further away in the network. The higher , the more strength of schedule is taken into account relative to immediate win percentage:

In this exponentiated matrix, the cell at row A and column B represents the relative strength of team A over team B based on the number of paths between them, counting shorter edges with more weight than longer ones. If we compare the strength of A over B to the strength of B over A, then we can get a good prediction for a potential game between them.

By summing the rows and columns of this exponentiated adjacency matrix, we can also get a sort of weighted “win percentage” for each team which accounts for a team’s record as well as opponents’ records, opponents opponents’ records and so on.

A latent issue with this weighting however, is that games are not played at random. In college football, for example, there are divisions in which every team must play every other team. This means that in a weaker division, a team could have a very strong win percentage- and could have beaten teams with a strong win percentage, yet all of these wins have come from inherently weaker teams. In the context of the exponentiated matrix, the cells between teams in common divisions will have much higher values, because more games are played between them, their opponents, and so on. Thus, these cells will have more weight on their calculated “exponential win percentage.”

The solution to this bias is to place more emphasis on the games between divisions, or more generally, to place more emphasis on connections between groups of teams where few connections exist. This can be accomplished by using the following simple transformation:

With E’, the cells between any two teams will be given the same weight. We can now get a more accurate weighted win percentage using this matrix. We can think of this as a projected win percentage if this team were to play every other team in the entire league.

* 1. **Page Rank**

Let’s now look at a method common in the field of social network analysis- PageRank. This algorithm was developed by Google founders Larry Page and Sergey Brin in order to rank results of internet searches. Page and Brin wanted to examine the behavior of a web user clicking around in a network of websites relevant to a given topic. They modeled this network as a markov chain in which the adjacency matrix was modified to provide the probabilities of a user moving between each pair of websites (called a Google matrix) incorporating a random probability of jumping from any page to any other page. They found that the dominant eigenvector of this google matrix gave the long term probability of arriving at each website.

In the context of this problem, the dominant eigenvector of the google matrix will be somewhat tantamount to predicting the winner of a tournament between the network of teams.

Win =Dominant Eigenvector of

Lose=Dominant Eigenvector of

Domination Score= Win/(Win+Lose)

* 1. **Elo Rating**

Another algorithm I want to explore is actually unrelated to graph analysis and was initially developed by Arpad Elo for rating chess players. The algorithm is based on the idea that after each game is played, we can adjust the rating of each player based on the outcome of the match and the prior ratings of each player. For example, if a player engages in a match with an opponent, their score will be updated according to the following equation:

* is 1 for a victory and -1 for a loss (usually elo incorporates draws as well, but they are rare or non-existent in our data sets so we will ignore them)
* is a factor of importance whereby games involving highly rated players will be counted less- i.e. highly rated players already have a well-established rating and should be less affected by a win or loss
* is an adjustable factor which describes how easily players can move up and down in rating (I’ll set it at 400)
* The rating of your opponent is their rating at the time of the match
* Every player will start out at 1450, and have a minimum score of 1015

This algorithm is good for rating players whose relative skill is constantly changing. Over time, different players can rise and fall in the ratings based on recent success. However, a setback to Elo is that it takes several games to calibrate. So, for season play where teams are relatively constant over the season and then undergo changes in the off-season- Elo may not be the best choice.

Let’s say that in the first game of a 16 game season, I beat a talented team which will go on to be 15-1. Since at the time of our match, we did not have any insight into the quality of this opponent, I will only get credit for beating an average team. In other words, we have no way to account for information we have learned about my opponents *after* I play them. This often has the effect of shrinking an already limited sample set.

* 1. **Iterative Power Rating**

In order to abate the shortcomings of Elo, I created an algorithm I call “iterative power rating” which iteratively adjusts the strength of a team by weighting victories and losses by the power score of each team played. Formally, we follow the procedure below-

1. Set the power score of all teams to their win percentage for the first iteration
2. For all games, find the power score of the winner and loser. Adjust the power score of each team to not count the current game (i.e. for a given game, we want to know the power score of all games outside of this one):

For the game’s winner:

For the game’s loser:

1. For each team, sum the power score of all opponents beaten to find wP:
2. For each team, sum the power score of all opponents lost to, in order to find lP:
3. Set the new Power score to
4. Normalize all Power scores to the range [0,1]:

At this point, you may be asking (reasonably) - what guarantees that this algorithm will converge on a solution? Following the path of PageRank and HITS algorithms, we should seek to relate the iterative algorithm to a linear algebra result to ensure convergence. The truth is that I’m not good enough at linear algebra to give you a formal proof of a relationship- but I can start you down the path of one, and show you a couple of examples of the algorithm converging in real data.

Let’s expand the first iteration (we will leave out the adjustments in step 2 to make the math cleaner, and only slightly alters the end result)

This is simply the sum of all wins by my opponents beaten divided by the sum of all wins and all losses by my opponents beaten. Described in terms of the adjacency matrix of the graph, A-

This is simply the sum of all losses by my opponents lost to divided by the sum of all wins and all losses by my opponents lost to. Described in terms of the adjacency matrix of the graph, A-

So the Power score looks like this:

* 1. **Collaborative Filtering**

Collaborative filtering is a popular method among recommender systems. Take NETFLIX movie recommenders for example- the idea is to take users with similar tastes to you and use their ratings of a movie to predict how you will likely rate a given movie. In our case, we will be looking at the performance of teams so the formulation is slightly different.

### 3 Data Sets

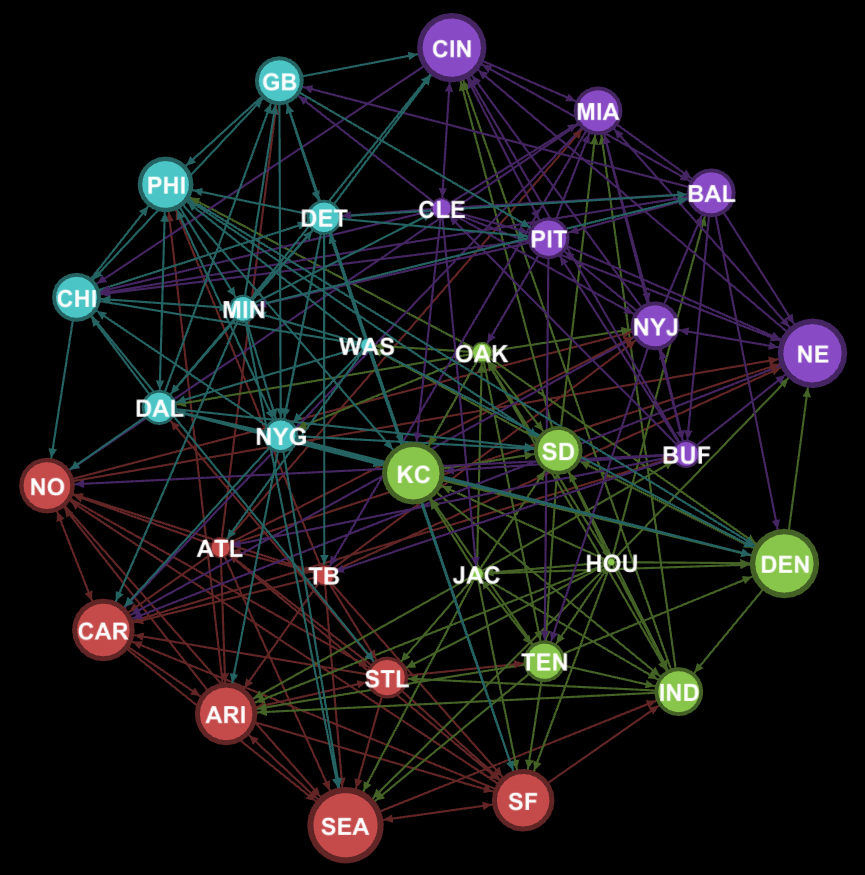
**NFL:** Regular and post season NFL games since 2000, bought from Armchair Analysis. Additional data includes score of each game and home/visiting team.

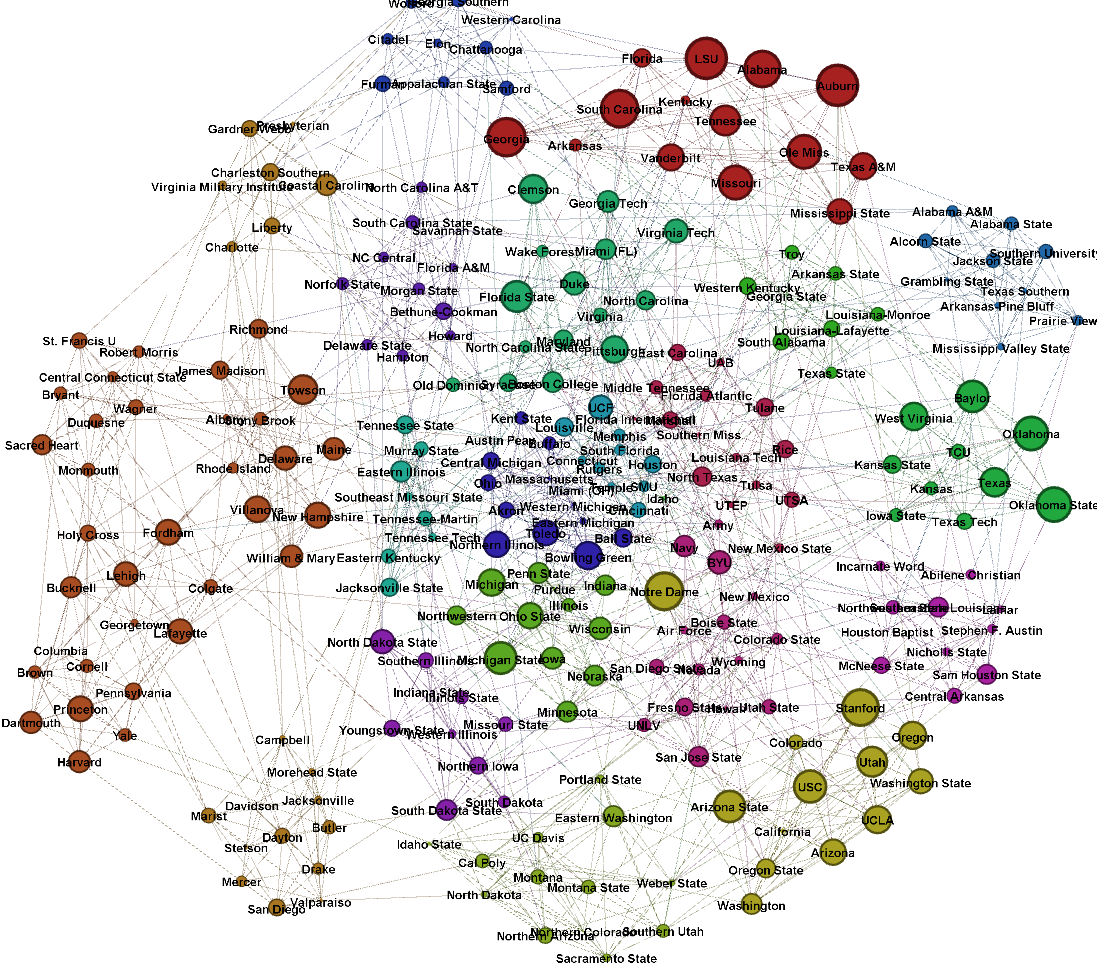
**NCAAF:** Division I NCAAF regular season and bowl games since 2002 scraped from ESPN.com. Additional data includes score of each game, division of each team, home/visiting team. The data contains some teams which are not Division I

**UFC:** The primary data which we seek to understand is a log of all major UFC fights since 1995. Also included is a log of other non-UFC mixed martial arts fights which we can use to further enhance our understanding of the fighters engaged in UFC. Unlike the other two data sets, the UFC fights will not be broken down by season.

Below is a directed graph for samples of each of these data sets.

**Figure 1.1 NFL 2013 Season**



**Figure 1.2 Division I NCAA Football 2013 Season**

Below is a set of summary descriptive statistics about each data set. A notable measure of connectivity is number of games in a season divided by the total possible games if each team played each other team one time. This is given by the equation where E is the number of edges and n is the number of nodes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Set | Years | Edges (Matches) | Nodes (Teams/Fighters) | Edges/Node | Density |
| NFL | 2000-2013 | 3,722 | 32 | 8.34 | 26% |
| NCAAF | 2002-2014 | 18,124 | 494 | 3.06 | 7.4% |
| UFC | 1995-2013 (will be considered one season) | 23,697 (All)  2,772 (UFC) | 12,270 (All)  1,269 | 1.93  2.18 | .17%  .016% |

### 3 Rating Effectiveness

To measure the success of these

Predictive Power will be measured in three ways

### 4 Formulating the Issue- Why don’t we do better

Before diving into real data, I’ll formalize the problem that we want to solve in the context of a few artificial networks. I also want to examine a few key assumptions we will need to make and what will happen when these assumptions are weakened.

**Transitivity of Wins**

Any model based on prior win/losses and strength of schedule will have to assume that team dominance is *strictly* *transitive*. That is to say, if team A beats team B and team B beats team C, then it is more likely that A will now beat C. If this effect was pure, then we could model each team based on some underlying level of strength S. We could then construct an ordinal ranking of teams 1-i:

In this system, the winner of each game would simply be the team with the higher underlying value for S. Thus, while the full underlying ranking is unknown, each game gives us a useful inequality. Let’s say that we have four teams in a league, and they have played three games between them, giving us the following three inequalities

Team 2 beat team 1

Team 4 beat team 3

Team 3 beat team 2

We could now construct the full ordinal team ranking: , and perfectly predict the outcomes of any future games between these teams. Even without fully solving the system, we could use our knowledge of probability to make some useful projections. For example, let’s say that we only had the first two games above () and we want to find the probability that will beat . We have now constrained the ordering of our teams to the following possibilities:

[1]

[2]

[3]

[4]

[5]

[6]

Note that in 5 of the 6 possibilities is ranked ahead of , so the probability that will win is . We can use this strategy to predict the probability of outcomes in any game- whether or not each team has even played a game. However, we know our underlying assumption of strict transitivity to be wrong. Teams’ performance can vary over time due to matchups, weather, injuries, random chance, and a host of other reasons.

Let’s examine the performance of our prior algorithms in the context of this “perfect world.” Notice that raw win percentage actually still performs very well on games which it is able to classify- the flat line for each of the algorithms is simply the set of ties. Exponential win percentage and power are a little better adjacency performs perfectly

**Adding Randomness**

Let’s add a little bit of complexity to our model. Instead of a constant number, let’s model the strength of each team in a given game as a normal random variable with different means, but equal variance. represents the jth sample taken from the ith team:

Now, each sample represents a team’s performance during a given game based on an underlying assumption about the general strength of that team (although the measure is still unit-less). Here we are simply looking to properly order the means of these distributions

Let’s bring back our four team league with the same three games, this time incorporating modeling the performance in each game with a random variable

Team 2 beat team 1 at home

Team 4 at home beat team 3

Team 3 at home beat team 2

One way to look at this information is to estimate the parameters of each distribution by maximizing the likelihood of this combination of events. In other words, we want to solve the optimization problem

Where represents a vector of ’s. Let’s re-write the objective function in terms of the parameters and in the standard normal CDF.

By choosing our vector µ to optimize this likelihood function, we can get the ranking of teams which best explains the series of games so far. Now we just have to validate the primary assumptions of the model, namely the distributions chosen for team performance

We could easily eliminate our assumption of equal variance and reformulate the problem with a vector, , describing the variance of each team’s performance. However, this gives us nearly twice as many variables to solve for, at some cost to the accuracy of estimating each parameter. Another potential extension is to use the actual win margins of games to formulate games into equations rather than inequalities.

**Home Field Advantage**

Although our second model of team strength is clearly better than the first, it’s still easy to see some serious flaws. First, it is unclear that our choice of a normal distribution for a team’s performance is reasonable-and it is nearly impossible to test what distribution would be the best given the complexity of the problem. We also can be thrown off by trends such as home field advantage

Team 2 beat team 1 at home

Team 4 at home beat team 3

Team 3 at home beat team 2

**Team Segmentation**

### 5 Measuring and Adjusting Real Data

**League Parity**

**Team Momentum (Time Weighting)**

## 6 Conclusions