DS210 Final Project Report
By Seokhoon Shin

Analyzing the Premier League Network (2006/2007 to 2017/2018)

Dataset: Premier League Results and Stats

"results.csv"

1. Introduction

As a passionate soccer enthusiast and data analysis enthusiast, I embarked on a fascinating journey to explore the dynamics of the Premier League. This project involves analyzing a dataset spanning the results and statistics from the 2006/2007 season to the 2017/2018 season. The Premier League has always captivated football enthusiasts worldwide, and this dataset provided an excellent opportunity to apply network analysis techniques using Rust. The goal was to compute centrality measures for the graph composed of Premier League teams as nodes and matches (The statistical relationship between teams : wins, losses, and goals) as edges. The project merges my love for soccer with my interest in data analysis, aiming to unveil hidden patterns within the league's structure that may not be immediately apparent through traditional statistics.

2. Project Overview

In this project, I designed a clear structure with several key components:

- main.rs: Serving as the entry point, this module is the foundation of the project.
- network_centrality.rs: This module houses functions for computing betweenness centrality using Dijkstra's algorithm.
- graph_construction.rs: It includes functionality for constructing the graph structure from the dataset.
- I imported essential components from the petgraph crate, a Rust library for graph data structures and algorithms.

3. Network Analysis Components

network_centrality.rs:

- calculate_betweenness_centrality: This function calculates betweenness centrality for each node in the graph using Brandes' algorithm.

- find_shortest_paths: It finds the shortest paths from a given start node to all other nodes using Dijkstra's algorithm.

graph_construction.rs:

- In this module, a TeamStats struct is defined, encapsulating various fields pertaining to a football team's seasonal statistics, including the team's name, wins, losses, goals, and the season. This struct is equipped with a method, calculate_edge_weight, which establishes the criteria for the weight of edges within the graph. The weights are calculated based on the absolute difference in the number of wins between teams, reflecting the competitive distance between them.

4. Analysis Insights

A pivotal finding from this analysis is that Swansea City surfaced as the team with the highest betweenness centrality score of 5304.176460496238. This indicates that within the network of Premier League teams, Swansea City occupies a highly influential position. Such a substantial centrality score suggests that Swansea City plays an unexpected yet central role in the league's structure, significantly influencing the season's dynamics. Even teams like Stoke City and Watford, which also exhibit high centrality scores, demonstrate their considerable roles within the network. These insights challenge the traditional view that a team's influence is solely a product of their media presence or historical success. Instead, they reveal the strategic importance of a team's position within the network of matches. In the competitive landscape of the Premier League, it is clear that teams such as Swansea City can be instrumental in shaping the narrative of the season, just as much as the traditionally dominant clubs. This analysis underscores the importance of network theory in sports analytics, providing a deeper understanding of the interconnectedness that defines the Premier League.

5. Dijkstra’s Algorithm for the Dataset

I chose to use the Betweenness Centrality measure algorithm based on Dijkstra's shortest path algorithm for this project for several reasons:

- Weighted Graph: The dataset includes match results with different outcomes (wins, losses, draws), making it essential to consider edge weights. Dijkstra's algorithm naturally supports weighted graphs.
- Importance of Connections: The Betweenness Centrality measure helps identify nodes (teams) with a significant impact on the graph's connectivity. Teams with high betweenness centrality can have a crucial influence on other teams' performance and connectivity within the league.

- Shortest Paths: To calculate Betweenness Centrality, we need to find the shortest paths between all pairs of nodes. Dijkstra's algorithm efficiently computes these paths, especially for sparse graphs like the Premier League network.
- Complexity: Despite the dataset containing results from 4560 Premier League matches (380 matches across 12 seasons from 2006/2007 to 2017/2018), equating to 9120 edges, and 9121 nodes, Dijkstra's algorithm remains effective with a complexity of O(18241*log(9121)).
- Directionality: Premier League matches are directed (team A vs. team B), and Dijkstra's algorithm handles directed graphs effectively.

6. Output - Betweenness Centrality

```
    Finished dev [unoptimized + debuginfo] target(s) in 1.13s
     Running `/Users/jasonshin/Desktop/plstats/target/debug/plstats`
Team: Swansea City, Centrality: 5304.176406496238
Team: Stoke City, Centrality: 4825.497407934054
Team: Watford, Centrality: 3356.8604874807706
Team: Newcastle United, Centrality: 2984.3566792710412
Team: Sunderland, Centrality: 2014.459465429045
Team: Crystal Palace, Centrality: 1535.8172334159392
Team: Middlesbrough, Centrality: 1430.2654349560503
Team: Hull City, Centrality: 1395.4036747129226
Team: Queens Park Rangers, Centrality: 1312.2386370510194
Team: West Ham United, Centrality: 1223.8988577628052
Team: Brighton and Hove Albion, Centrality: 1160.785670962222
Team: Southampton, Centrality: 1148.6267767109289
Team: Blackburn Rovers, Centrality: 1120.6344281671813
Team: Norwich City, Centrality: 989.9753702082471
Team: Huddersfield Town, Centrality: 939.1846093011893
Team: West Bromwich Albion, Centrality: 934.6978006878711
Team: Fulham, Centrality: 880.2751998991872
Team: Everton, Centrality: 786.1785509036375
Team: AFC Bournemouth, Centrality: 782.2004412330433
Team: Leicester City, Centrality: 745.9521387474771
Team: Wigan Athletic, Centrality: 707.1207930393481
Team: Aston Villa, Centrality: 613.0485277703567
Team: Arsenal, Centrality: 538.8448812288478
Team: Birmingham City, Centrality: 537.4282732548771
Team: Portsmouth, Centrality: 409.12606145515446
Team: Bolton Wanderers, Centrality: 404.72675829272134
Team: Burnley, Centrality: 385.84345068326974
Team: Chelsea, Centrality: 379.3788099764335
Team: Liverpool, Centrality: 329.51538951233806
Team: Manchester United, Centrality: 276.68873181815746
Team: Tottenham Hotspur, Centrality: 252.91979483226518
Team: Cardiff City, Centrality: 225.72366685266073
Team: Manchester City, Centrality: 202.9277100730605
Team: Reading, Centrality: 176.76448788170507
Team: Wolverhampton Wanderers, Centrality: 143.13364790782074
Team: Charlton Athletic, Centrality: 119.04787824987183
Team: Derby County, Centrality: 83.49979302670138
Team: Blackpool, Centrality: 66.98052860319184
Team: Sheffield United, Centrality: 27.86648480992664
```

Betweenness centrality measures the extent to which a node (in this case, a football team) stands on the shortest path between other nodes. A node with high betweenness centrality has a greater 'control' over the network because more shortest paths pass through it. Teams like Swansea City, with the highest centrality score, seem to have a significant impact on the Premier League network. This suggests that they are involved in a large number of matches that are crucial for other teams to advance in the league, regardless of their position in the standings. Conversely, teams with minimal centrality scores, such as Sheffield United, appear to have matches that influence the league's structure to a lesser extent. Such insights are valuable for understanding the strategic importance of matches throughout the season beyond wins and losses, highlighting the interconnectedness of the league.

```
Graph Nodes:
Node Index: 0, Team: Manchester United
Node Index: 1, Team: Chelsea
Node Index: 2, Team: Liverpool
Node Index: 3, Team: Arsenal
Node Index: 4, Team: Tottenham Hotspur
Node Index: 5, Team: Bolton Wanderers
Node Index: 6, Team: Reading
Node Index: 7, Team: Blackburn Rovers
Node Index: 8, Team: Everton
Node Index: 9, Team: Portsmouth
Node Index: 10, Team: Middlesbrough
Node Index: 11, Team: West Ham United
Node Index: 12, Team: Aston Villa
Node Index: 13, Team: Manchester City
Node Index: 14, Team: Newcastle United
Node Index: 15, Team: Sheffield United
Node Index: 16, Team: Wigan Athletic
Node Index: 17, Team: Charlton Athletic
Node Index: 18, Team: Fulham
Node Index: 19, Team: Watford
Node Index: 20, Team: Sunderland
Node Index: 21, Team: Birmingham City
Node Index: 22, Team: Derby County
Node Index: 23, Team: Stoke City
Node Index: 24, Team: Hull City
Node Index: 25, Team: West Bromwich Albion
Node Index: 26, Team: Wolverhampton Wanderers
Node Index: 27, Team: Burnley
Node Index: 28, Team: Blackpool
Node Index: 29, Team: Norwich City
Node Index: 30, Team: Swansea City
Node Index: 31, Team: Queens Park Rangers
Node Index: 32, Team: Southampton
Node Index: 33, Team: Crystal Palace
Node Index: 34, Team: Cardiff City
Node Index: 35, Team: Leicester City
Node Index: 36, Team: AFC Bournemouth
Node Index: 37, Team: Brighton and Hove Albion
Node Index: 38, Team: Huddersfield Town
```

7. Conclusions

Utilizing Dijkstra's algorithm to compute Betweenness Centrality, this project revealed the covert pivotal forces within the Premier League. Swansea City's top centrality score was a remarkable discovery, indicating their substantial influence on the league's network dynamics. This insight is crucial for forecasting potential league developments and emphasizes how teams can impact the league's intricate web of connections beyond their immediate match outcomes. This project underscores the significance of considering the broader network effects and demonstrates the profound capabilities of network analysis in sports analytics.

8. Test

```
    Finished test [unoptimized + debuginfo] target(s) in 0.67s
     Running unittests src/main.rs (/Users/jasonshin/Desktop/plstats/target/debu
g/deps/plstats-e8341a5ba2c943d0)

running 2 tests
test tests::test_centrality_calculation ... ok
test tests::test_graph_construction ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; fini
shed in 0.00s
```

9. Resources

- petgraph Documentation
- DS210 lectures (28, 30)
- tempfile Documentation
- std::io::Write Documentation
- graphrs Crate
- Dijkstra's Algorithm on Wikipedia
- petgraph::prelude Documentation
- petgraph::graph::Graph Documentation

10. Acknowledgments

I would like to express my gratitude to the creators of the Premier League dataset, the Rust programming community, and the instructors of DS210 for their valuable resources and support throughout this project.