Problem Chosen
D

2020
MCM/ICM
Summary Sheet

Team Control Number
2006068

Summary

Cooperative strategies play an indispensable role in the increasingly connected society, which contains the power of wisdom and shines brightly. The structural strategy in football is a magic weapon for a team to keep winning. It can be seen that scientific and creative analysis of a team's cooperation strategy has unquestionable significance for the future of the team.

First, we establish a network analysis model, which is analyzed at three levels: overall, local, and individual. We explore the overall dynamics of the entire game and the entire season of the team, as well as the interaction between the players on the field.The model vividly demonstrate the position of the players in the game and the trajectory of the pass. And the team's victory are quantified,to analyze the winning magic of team victory.

Secondly, we have established a team network performance evaluation model, adopting the analytic hierarchy process to determine performance indicators which reflect team success. The influential factors, and essence of team strategy are further researched to evaluate team performance.

In addition to team cooperation on the field, the formulation of pre-match strategies and the starting players also have a great impact on the outcome of the game. Since we cannot directly see the coordination between players or the distribution of contributions in the entire passing network system, we conduct a micro-analysis of the local network. In the local network, we use the condensed Agglomerate subgroup analysis which can simplify the complex overall passing network and make it easier for us to find the cooperation contained in the football network strategy.

In order to avoid mistakes in the model, we have established a validation model to ensure that there are no uncertain factors in the model that affect the analysis results. Finally, after ensuring that the results we analyze are true and effective, we make strategic recommendations to the Huskies team based on the conclusions obtained and develop a general team performance based on an in-depth analysis of the team strategy model.

Key words: Network analysis model ;Agglomerate subgroup analysis ;AHP method Cooperative strategies

Contents

1. Introduction

1.1 Background

As society becomes more interconnected, the success of teamwork plays an unquestionable role in various fields. On the one hand, the overall performance and coordination of the team determine the core strength of the team. On the other hand, the cooperation and communication between the team members in the team can reflect the flexibility of the team's operation. This is also the case in football matches. The drastic game depends not only on the core leadership of the team, but also on the overall strategic cooperation of the team. The cooperation of players in a game is a social network, which analyzes the team's tactics and structure strategy through the network. Therefore, in order to meet the needs of the Huskies coach, it is important to establish a model to analyze the football team's network and structural strategy. Based on previous studies, we improved and created microscopic macro network analysis models and structural decision models to better analysis of the team's "cooperation".

1.2 Restatement of the Problem

In response to Coach Husky's request and exploring team dynamics throughout the game and season to help determine specific strategies that can improve teamwork for next season, our team will address the following issues:

●Establish a network for passing between players, use the transfer network to identify network patterns, and explore the characteristics of the team's overall network and local network.

●Determine performance indicators that reflect the success of teamwork and come up with the best strategy to win, to capture aspects of team structure, configuration, and motivation.

●Use the insights we have gained from the teamwork model to tell the coach what strategies are most effective for the Husky team. According to our passing network analysis, and give the coach advice to change the team structure of the team next year, to achieve better results.

●The network passing model established by us and the results obtained through analysis. Look for the most important factors in teamwork. When faced with the difficulties of other teams, find out the best method we should adopt.

1.3 Overall concept

On a micro-scale, the importance of each player is related to it: feature vector centrality, a measure of importance obtained from the feature vectors of the adjacency matrix; intimacy, a measure of what a player must pass to reach any other player in the team The minimum number of steps; or intermediate, which indicates the number of times a given player needs to complete a route that connects any two other players on the team.

On a macro scale, you need to determine the position of each player, establish a passing network, and analyze the number of passes between players and the position of the player to control the field to quantify the probability of each game.

1.4 Overview of our work

First of all in order to clearly describe the dynamic process of the game from a micro perspective to a macro perspective, we built a network model using a social network analysis model, and a team model based on the binary. We gave a preliminary drawing of the passing network based on the data given. The directed graph allows us to see the passing relationship and passing trend of both sides more visually and intuitively, to judge the emphasis on offense and defense. **Secondly**, we use the analytic hierarchy process to build a performance evaluation model, and further find the best by comparing 14 wins of the Husky team Cooperate with the group and core players, and further determine which combination is more suitable for the Huskies. In analyzing the team's performance indicators, the structure configuration dynamically applied AHP and found the optimal plan.

On the basis of the performance evaluation model, we start from a micro perspective and build a local analysis model. In the local network, we use condensed subgroup analysis, which can simplify the complex overall pass network we build and make it easier for us to find The substructures and their relationships contained in the network, so as to obtain the best team structure strategy.

2. Social network model

2.1 Assumptions

●In this article we study the effects of teamwork or individual abilities on the game results. It is assumed that the influence of other factors on the game results is negligible, such as the environment, the player's psychological state and other factors.
●Assume that the time out has no effect on teamwork and matchmaking.
●Assume Referee bias and coach departure do not affect the game and Players were not injured during the game.

2.2 Notations

Table 1: Notations of the Social network model

Symbol	Definitions
Y	Y-axis
p_{ave}	Average serve position
d_{ave}	Team formation density
W	Win
T	Lose
L	Tie
c_n	Clustering coefficient
c_{ave}	Average clustering coefficient
N_{cr}	Actual number of neighboring nodes

N_{ca}	Maximum number of neighboring nodes
d_c	Number of neighboring nodes
d	Average shortest path
p_{ij}	Topological distance
B	Judgment matrix
CI	Consistency testing standards
RI	Stochastic consistency index
CR	Consistency ratio
N	Maximum characteristic root
W_i	Degree of influence of the i-th factor
P	Criteria and decision-making weights

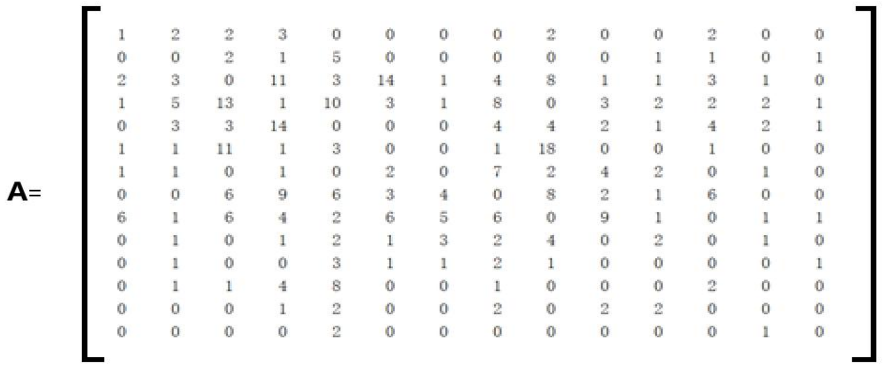
2.3 Adjacency matrix

The adjacency matrix is one of the representation structures of the passing network. In the adjacency matrix, the element A_{ij} in the i-th row and j-th column represents the number of times the i-th player passes to the j-th player in the entire game. The more passes between two directly connected players, the smaller the "topological distance" between them, and so does the cooperation between them. Therefore, **the adjacency matrix indirectly reflects the closeness of cooperation between players in the team through the number of passes.** If the reciprocal of the number of passes is used to represent the "topological distance" between players, then a topological distance matrix can also be obtained.

Taking the data given by the game title fullevents.csv as a sample, we imported the data into the mysql database, wrote a Java program to read and process (Appendix 1), and calculated the adjacency matrix **A** (Appendix 2) describing the passing network in all games.

For example, Figure 1 shows the adjacency matrix of the passing network of the 14 Huskies players (including substitutes) in the first game. Since the passing between players is one-way, we can build a directed graph network accordingly. The main diagonal element of the matrix is not 0, indicating that there are situations in which the player passes to himself.

Table 2:The adjacency matrix of the football passing network of Huskies in the match one



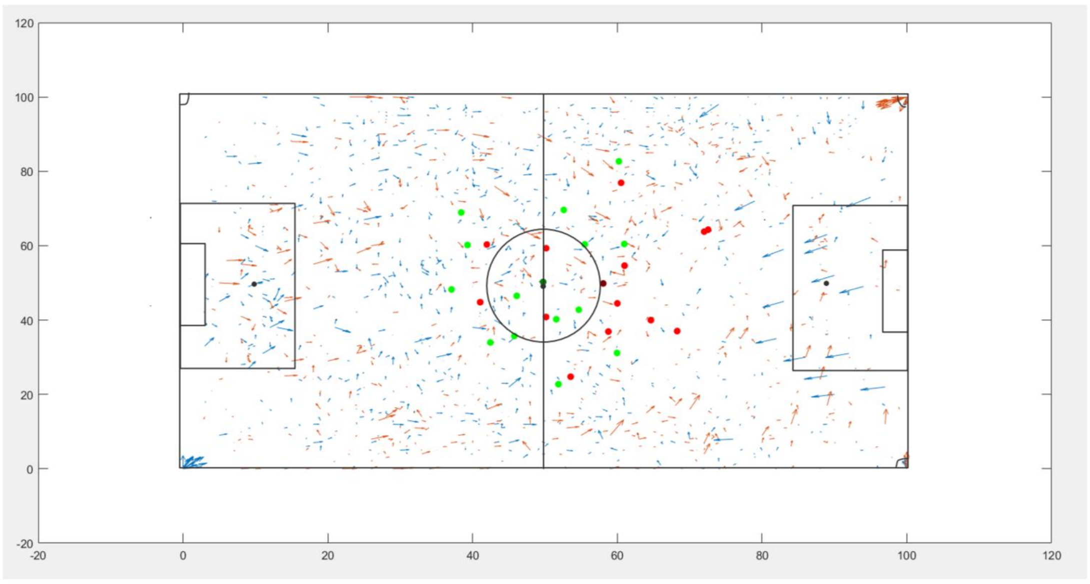
$$\mathbf{A}=\begin{bmatrix}
1 & 2 & 2 & 3 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 2 & 1 & 5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
2 & 3 & 0 & 11 & 3 & 14 & 1 & 4 & 8 & 1 & 1 & 3 & 1 & 0 \\
1 & 5 & 13 & 1 & 10 & 3 & 1 & 8 & 0 & 3 & 2 & 2 & 2 & 1 \\
0 & 3 & 3 & 14 & 0 & 0 & 0 & 4 & 4 & 2 & 1 & 4 & 2 & 1 \\
1 & 1 & 11 & 1 & 3 & 0 & 0 & 1 & 18 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 2 & 0 & 7 & 2 & 4 & 2 & 0 & 1 & 0 \\
0 & 0 & 6 & 9 & 6 & 3 & 4 & 0 & 8 & 2 & 1 & 6 & 0 & 0 \\
6 & 1 & 6 & 4 & 2 & 6 & 5 & 6 & 0 & 9 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 2 & 1 & 3 & 2 & 4 & 0 & 2 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 3 & 1 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 4 & 8 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 1 & 2 & 0 & 0 & 2 & 0 & 2 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

Pic 1 The adjacency matrix of the football passing network of Huskies in the match 1.

2.4 Center position and formation density

The adjacency matrix describes how tightly the team passes, but it cannot describe the geometrical structure of the team on the court. Obviously, the team's formation structure is very important for the game. According to the data, in the model we uniformly establish a coordinate system for the stadium, and the horizontal and vertical axis coordinate ranges are [0,100]. The zero point of the coordinate system is always relative to the research target team. If they face each other, the zero point of the coordinate system is at the right behind The apex of the stadium.
In a ball game, the formation of the team changes dynamically. As an example, we first draw the starting and falling points of the ball in all passing events in the first half of the first game:

Table 3:Full-field center position and formation density



The goalkeeper is not included in the picture. The blues are the passes of the Huskies team. The oranges are the passes of the opposing team. The green dots are the average serve positions of the Huskies players. The red dots are the opponent's average serve positions. Center, the dark green dot is the center of Huskies, the dark red dot is the center of the opponent. Since the coordinate system is always relative to itself, **both sides of the figure have their own goals on the left**. As can be seen, the passing distance in the figure is generally short.

The following calculates **the relationship between team formation characteristics and the outcome of the game**. Here we use the team's **center position** and **density** to summarize the formation characteristics. Suppose that the horizontal and vertical coordinates of team members' serve positions in a game are represented by matrices **X** and **Y**, respectively.

$$X = \begin{Bmatrix} x_{11} & x_{12} & \cdots & x_{1s_1} & 0 \\ x_{21} & x_{22} & \cdots & x_{2(s_2-1)} & x_{2s_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{n(s_n-1)} & x_{ns_n} \end{Bmatrix}$$

In the above formula, S_n is the number of serves of the n-th player, and x_{ij} represents the x coordinate of j-th serve of the i-th player. The number of rows in the matrix **X** is $\max\{s_1, s_2, ..., s_n\}$ If s_i is less than the number of rows in the matrix, the extra elements are filled with 0.

The average serve position is represented by the vector group p_{ave}:

$$P_{ave} = (X_a, Y_a) = \begin{Bmatrix} x_{1a} & y_{1a} \\ x_{2a} & y_{2a} \\ \vdots & \vdots \\ x_{na} & y_{na} \end{Bmatrix}$$

Where n is the number of team players, and (x_{na}, y_{na}) is the average serve position of the nth player, that is:

$$x_{na} = \frac{\sum_{i=1}^{s_n} x_{ni}}{s_n}$$

The center of the team is obtained by follows:

$$(x_{ave}, y_{ave}) = (\frac{\sum_{i=1}^{n} x_{ia}}{n}, \frac{\sum_{i=1}^{n} y_{ia}}{n})$$

For the formation density of the team, we use the average distance between the player's average serve position and the center for evaluation. The greater the density d_{ave}, the smaller the average distance.

$$d_{ave} = (\frac{\sum \sqrt{(x_{na} - x_{ave})^2 + (y_{na} - y_{ave})^2}}{n})^{-1}$$

In the appendix code, we use HQL(Hibernate query language) and Java to calculate the center point of each game team and draw the following figure:

Table 4:The difference between the x coordinate of the team center point and the field center point

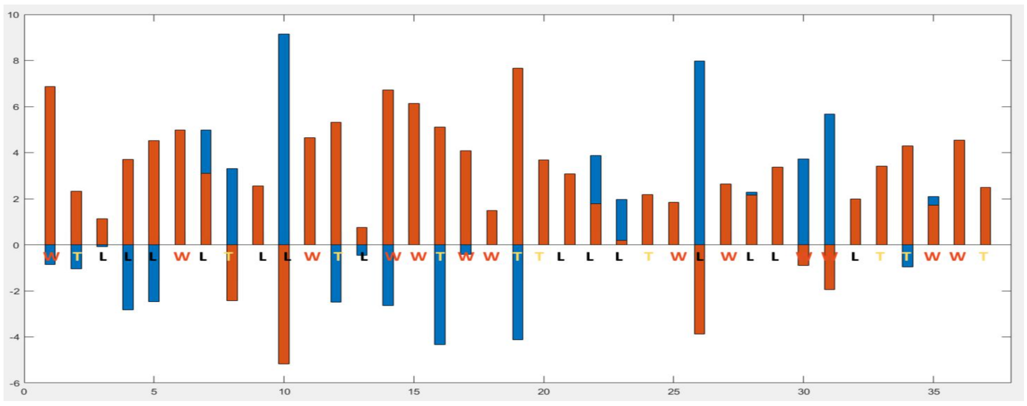
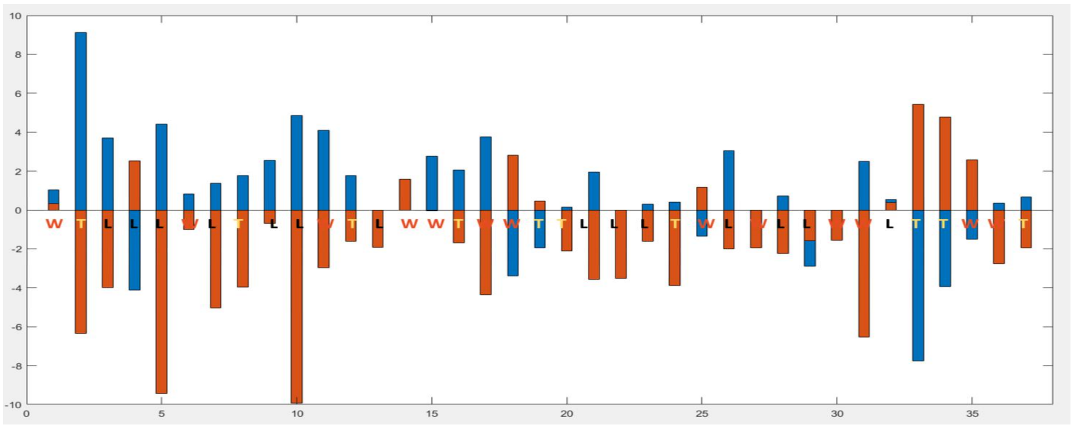


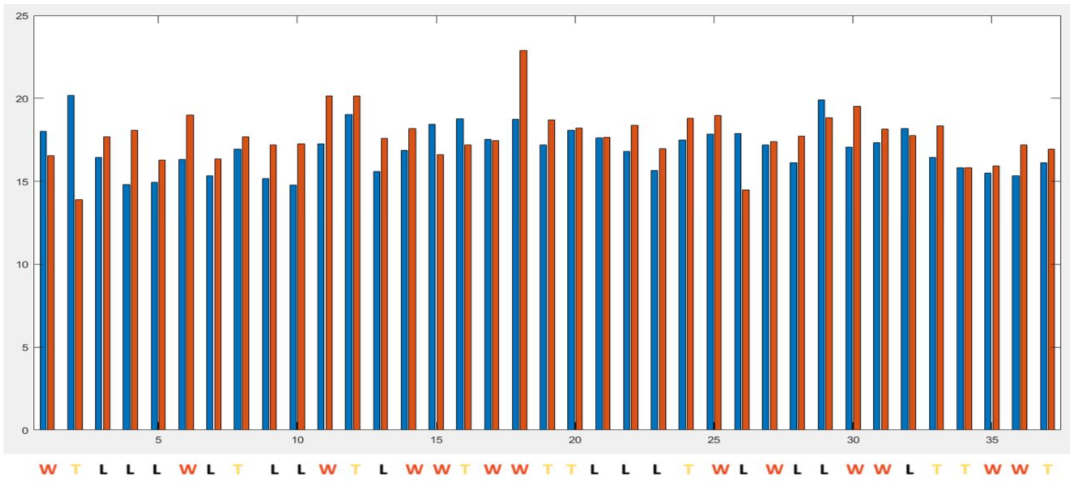
Table 5:Difference between the y-coordinate of the team center point and the field center point



In the above two figures, the horizontal axis is the number of matches, W, T, and L represent wins, losses, and ties, orange is the opponent, and blue is Huskies. The vertical axis of the first picture represents the difference between the x-coordinates of the team center point and the field center point, and the vertical axis of the second picture represents the y-coordinates of the team center point and the field center point.

In addition, we also calculated the average distance of all team players from the center in 38 games:

Table 6:Average distance of players from all teams to center in 38 games



In the third picture above, orange is the opponent, blue is the Huskies, the horizontal axis is the number of games, and the vertical axis is the average distance between the player and the center, which describes the density of the team formation. Analyzing this graph separately, we can find that the average distance between the player and the center in all games is about 17 and the fluctuation is small.

Combining the first picture and the third picture, we can roughly see whether the team prefers offensive or defensive strategies in the game. A regular bar indicates that the players in the entire game are generally biased towards offense, and vice versa. But neither excessive offensive strategy nor excessive defensive strategy is conducive to winning. The Huskies lost their excessive offense in their 10th and 26th games, while maintaining the balance of offense and defense in the 14, 15, 16, 17, 25, 27 games. However, the horizontal distribution the second chart shows has almost no regularity.

2.5 Clustering coefficient

Through the adjacency matrix description of the graph, we can analyze and calculate the team's cooperation degree in each game. The **clustering coefficient** [2] is a parameter used to describe the degree of team cooperation in the social network analysis model.

The clustering coefficient of a node is called **the local clustering coefficient**. First calculate how many edges may form between all nodes connected to the current node as the denominator, and then calculate how many edges actually have as the numerator. It can be expressed as:

$$c_n = \frac{N_{cr}}{N_{ca}}$$

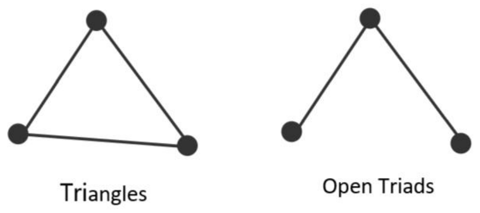
In the formula, N_{cr} is the number of edges actually formed by adjacent nodes, and N_{ca} is the number of edges that may be formed by adjacent maximum nodes. Obviously, its relationship with the number of adjacent nodes d_c is:

$$N_{ca} = C_{d_c}{}^2 = \frac{d_c(d_c - 1)}{2}$$

The C above is a combination number sign. So the clustering coefficient of a node is:

$$c_n = \frac{N_{cr}}{\frac{d_c(d_c - 1)}{2}}$$

The average clustering coefficient of the network is the average of the clustering coefficients of each node in the network. Another calculation method is to divide the number of balanced triangles in the figure by three times the number of open triads. This method is called Transitivity.



$$c_{ave} = \frac{3n_{triangles}}{n_{opentriads}}$$

However, when the network is weighted, we cannot simply calculate the number of nodes connected between them, and also determine how the link weights are distributed. In this case, the number of transfers between them is not constant. Therefore, we use **weighted clustering coefficients**. In the passing network, weighted clustering coefficients describe the team's tendency to form a balanced triangle between players.

The method we use is to multiply the local clustering coefficient mentioned above by the degree of the node and then average it.

$$c_{ave} = \frac{\sum c_n d_c}{n}$$

Using the data given in fullevents.csv, the weighted clustering coefficient was calculated using the obtained adjacency matrix (Appendix 2):

Table 7:Huskies weighted clustering coefficient

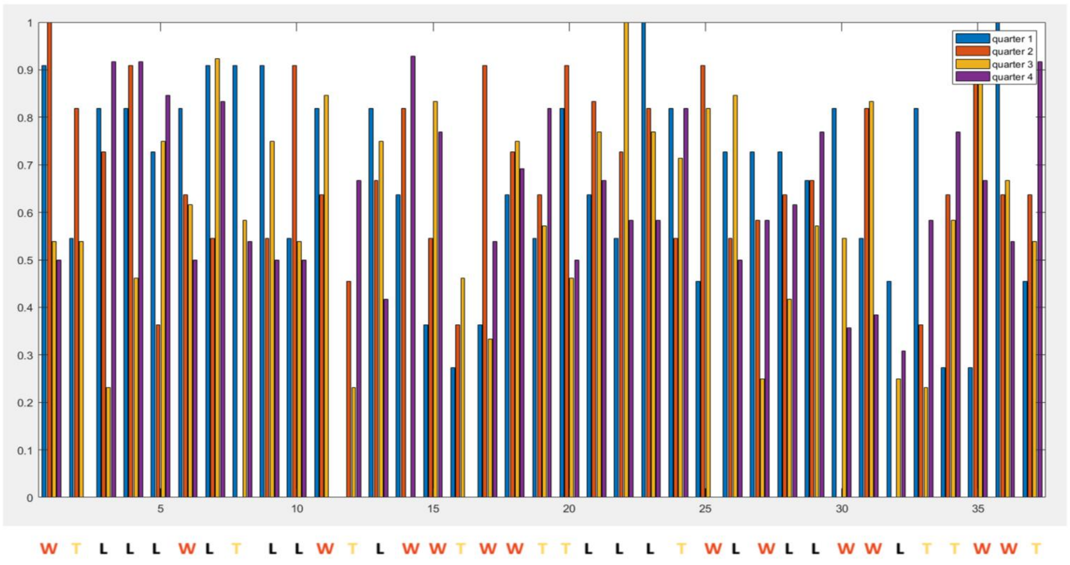
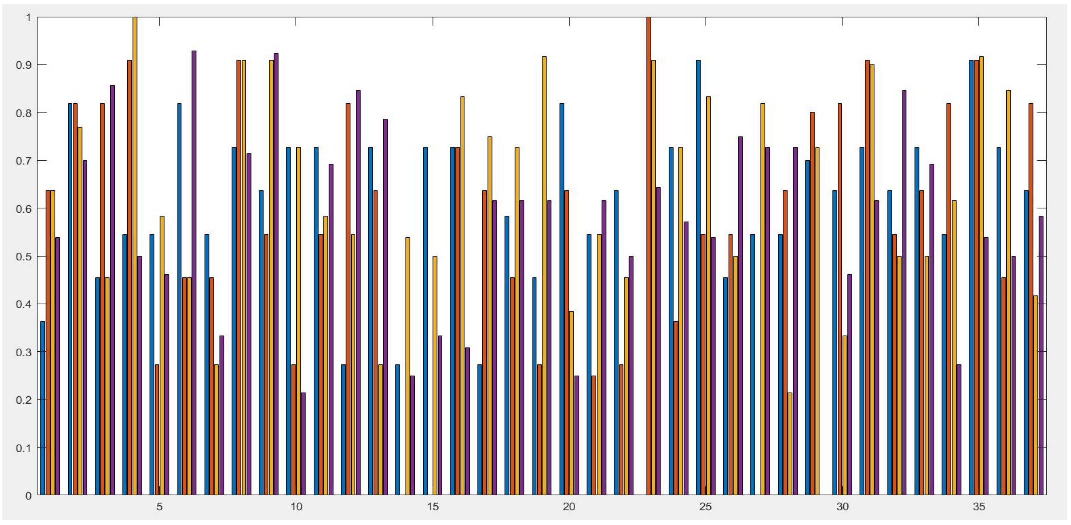


Table 8:Adversary weighted clustering coefficient



The two figures are the weighted clustering coefficients of the Huskies and the opponent. The horizontal axis is the number of games. In order to see the effect of time, we divide each game into 4 parts and calculate the weighted clustering coefficients. From the data we can see that **in many games, the side with the highest clustering coefficient often wins**. For example, in the first game, the average clustering coefficient of the Huskies was significantly higher than the opponents and eventually won; but there are still some games in which the game outcome are opposite to the performance of the clustering coefficient. For example, in the third game, the average clustering coefficient of the Huskies was also higher than that of the opponent team, but it lost. This shows that the clustering coefficient has a certain effect on judging the performance of a team, but due to the influence of other factors, it cannot be used as a single reference quantity.

2.6 Graph robustness

The graph robustness is whether the connectivity of the graph can still ensure connectivity if any nodes or edges are removed. [2] This is also very important for the passing network. We use the node strain energy sensitivity [3] to measure the stability of the entire passing network. The strain energy sensitivity of a node is the relative partial derivative of the node coordinates in the network. When the value of the node's strain energy sensitivity is greater, it means that the smaller the initial deviation of the node, the greater the load on the entire tree structure influences.

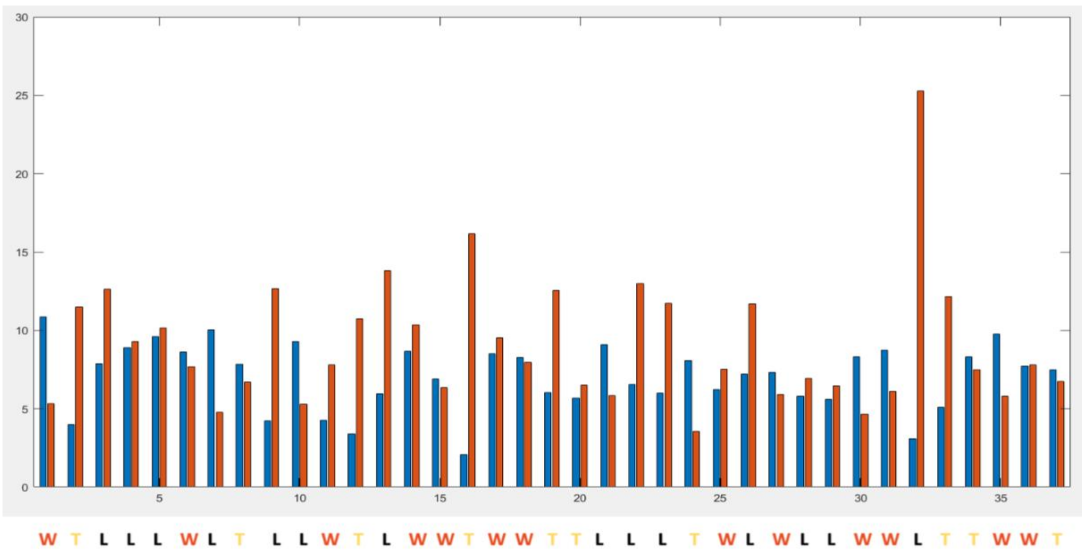
$$\alpha = \frac{\partial E}{\partial x}$$

Where E is the nodal strain energy, and x is the nodal coordinates. The concept of nodal strain energy is derived from elastic mechanics, which represents the energy stored when a material is strained. Networks and graphs also have the recovery properties of elastic materials.

Another quantity describing the stability of the graph is the eigenvalue λ1 of the adjacency matrix. As a rule of thumb, a network with a higher number of links (through) will have a higher λ1 and the network and the important nodes connecting them (Referred to as a matching network) will also have a higher λ1 [4].

The maximum eigenvalues of the adjacency matrix of the two teams in 38 games are calculated as follows:

Table 9:The influence of the maximum eigenvalue on the outcome of the game

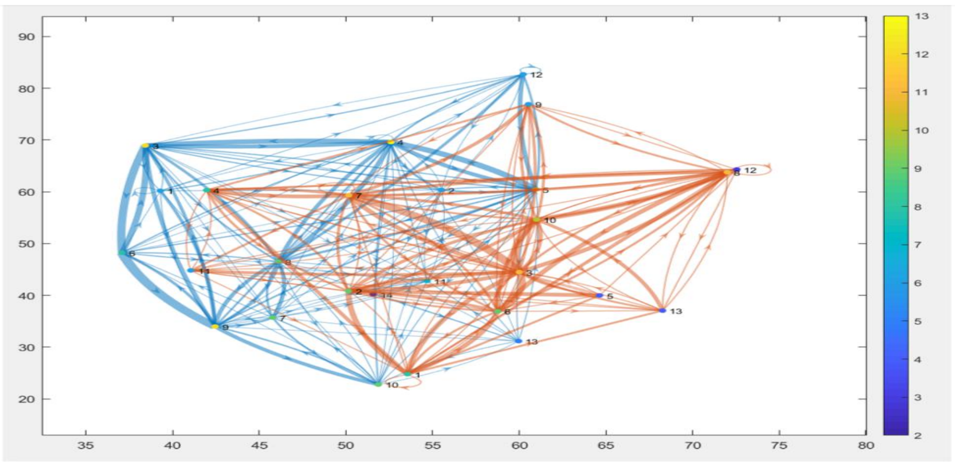


The influence of the maximum eigenvalue on the outcome of the game can be seen from the figure.

2.7 Directed map and the average shortest path

Based on the adjacency matrix and the average geometric position of the team, we can draw a directed graph of the passing network[6], and graphically describe the placeholders in a game. Using Java engine of matlab (Appendix I), we draw a directed map of the Huskies and the opponent team in the first game:

Table 10:The football passing network of the match

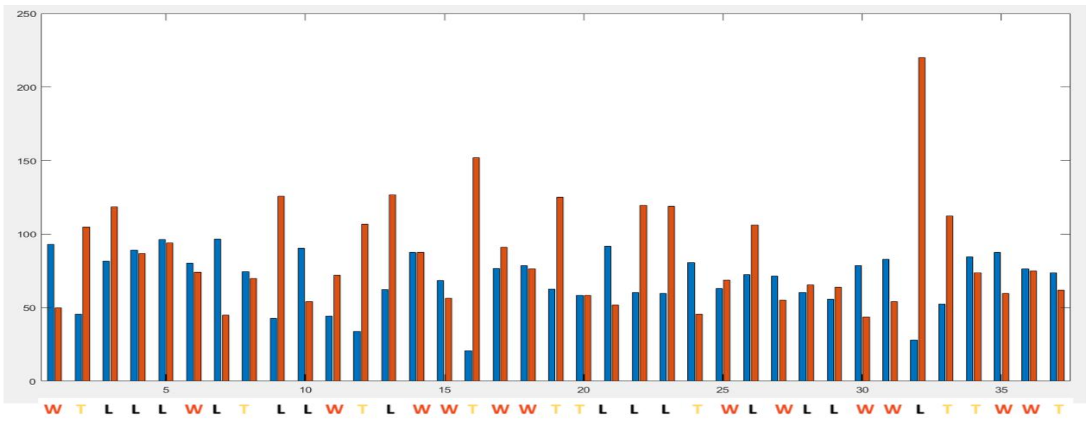


Pic2 The football passing network of the match 1.
The blue lines stands for passing among the team Huskies, while the oranges stands for the opponent team.

In this directed graph, each node represents a player, and the arrow line indicates that the player’s effective passes to other teammates. The more the passes, the thicker the line. The node color indicates the out-degree of the node. The node position is the average position of the player's passing point in this game. It can be seen that most of the Husky team's thick line is located in the backcourt position, and the opponent's thick line is often located in the front and midfield, which means that the Huskies is more focus on defense in this game. From the passing network diagram, you can see the passing relationship and passing trend of the entire team more intuitively. You can also roughly understand which nodes and connections in the team are more important, but you must understand to study the passing network characteristics, a quantitative analysis of the passing network is required.

First, we noticed that **the total number of passes** is critical to the connectivity of the network. Here are the total number of passes and their wins and losses in each game:

Table 11:the total number of passes and their wins and losses in each game



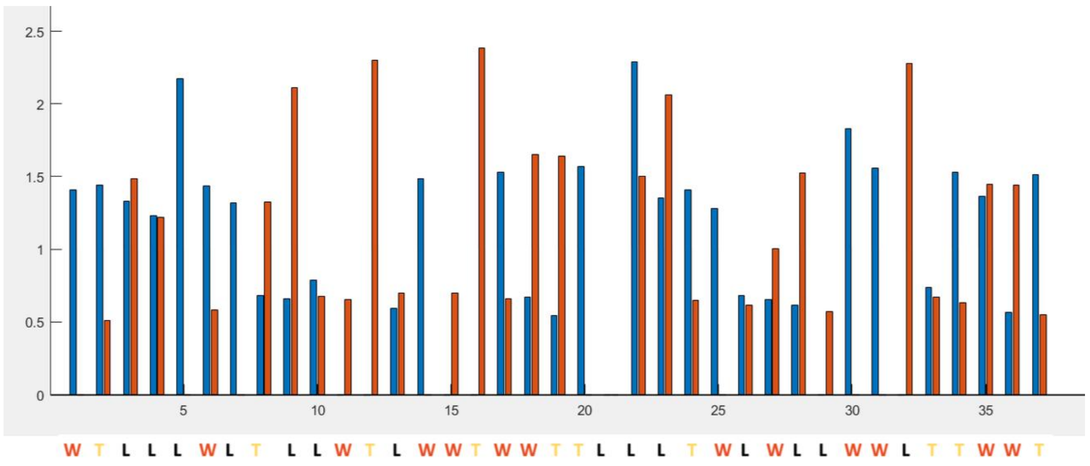
In the established passing network, there will be a shortest path for passing between any two players. The shortest path between two players is not necessarily a direct pass. It may be necessary to merge two or more optional paths into the shortest path between the two players. Because the number of passes between players is different, the passing network we set up is weighted, and the larger the weight, the shorter the topological distance between the two nodes established. In order to calculate the shortest path in the weighted graph, we use Dijkstra's algorithm for Pij and define the **average shortest path** *d*

$$d = \frac{1}{N(N-1)} \sum_{i,ji \neq j} p_{ij}$$

Where p_{ij} is the topological distance, the number of passes, and *N* is the total number of players in the team.

Here is the average shortest path between the Huskies and their opponents in 38 games:

Table 12:he average shortest path between the Huskies and their opponents in 38 games



In the figure above, the horizontal axis is the number of matches, the vertical axis is the average shortest path, orange is the opponent, and blue is the Huskies. In some cases, the two players may not be connected, which directly leads to the average shortest path of the entire team being positive infinity. These conditions are filtered out in the figure.

It can be clearly seen from the figure that the average shortest path of the team has a significant impact on the victory and defeat. The more the average number of passes, the shorter the shortest path and the higher the winning rate, but this is not absolute, such as the 15th opponent has a higher average number of passes, but the Huskies still win.

3.Team network performance evaluation model:

3.1 Symbols

Table 13:symbols of Team network performance evaluation model

Symbols	definition
B	Judgment matrix
CI	Consistency testing standards
RI	Random Consistency Standard
CR	Consistency ratio
n	Maximum characteristic root
Wi	Degree of influence of the i-th factor
P	Decision-level results

3.2 Establishment of the model

Since the team performance and level have the characteristics of multi-objective, multi-criteria, or unstructured characteristics, we use a qualitative and quantitative combination of a systematic and hierarchical AHP model to analyze the nature and influencing factors of complex decision-making problems. Based on in-depth research on its internal relationship, it uses less quantitative information to mathematicalize the thinking process of decision-making, thereby providing a simple decision-making method for multi-objective, multi-criteria, or complex decision-making problems without structural characteristics.

In order to analyze the team's best structural strategy, here we mainly start with 13 wins, and then analyze the best match structure of these 13 games. The goal is divided into three layers for analysis: the highest layer is the best structural configuration; the criterion layer is the number of goals scored, the number of goals conceded by the game, the average clustering coefficient of the team in the second half, the maximum feature of the team adjacency matrix, Five criteria for the average number of passes in the second half of the game; the decision-making layer is the 14 games to be selected, and the optimal alternative is extracted through hierarchical analysis, and the microstructure strategy and The team's tactical cooperation can finally capture the team's structure, configuration, and dynamic structural strategy, and analyze a reasonable and flexible team cooperation model.

Build a judgment matrix based on the weight given to the criterion layer：

$$B = \left\{ \begin{matrix} 1 & 2 & \frac{3}{2} & 2 & \frac{3}{2} \\ \frac{1}{2} & 1 & \frac{3}{4} & 1 & \frac{3}{4} \\ \frac{2}{3} & \frac{4}{3} & 1 & \frac{4}{3} & 1 \\ \frac{1}{2} & 1 & \frac{3}{4} & 1 & \frac{3}{4} \\ \frac{2}{3} & \frac{4}{3} & 1 & \frac{4}{3} & 1 \end{matrix} \right\}$$

By analogy, the judgment matrix between the criterion layer and the decision-making layer is constructed. Since the dimension of the decision-layer matrix is large, it can be queried in the appendix.

Then we introduced the consistency check standard:

$$CI = \frac{\lambda - n}{n-1}$$

Among them, when $CI = 0$ means complete consistency; when CI is infinitely close to 0, there is satisfactory consistency; the larger the CI, the more serious the inconsistency. In order to measure the size of CI, a random consistency index RI is introduced. The method is to randomly construct 500 pairwise comparison matrices $A_1, A_2, ..., A_{500}$, and obtain one-time indicators $CI_1, CI_2, ..., CI_{500}$.

The calculation formula of RI is as follows

$$RI = \frac{\sum_{k=1}^{500} CI_k}{500} = \frac{\frac{\sum_{m=1}^{500} \lambda_m}{500} - n}{n-1}$$

The consistency ratio is defined as follows:

$$CR = \frac{CI}{RI}$$

Generally, when the consistency ratio CR <0.1, it is considered that the degree of inconsistency of B is within the allowable range, and there is satisfactory consistency. After the consistency test, the normalized feature vector can be used as the weight vector, otherwise the matrix B must be reconstructed.

After consistency check, the judgment matrix meets the requirements. Then we naturally choose the normalized feature vector corresponding to the largest feature root n： $w=\{w_1 \quad w_2 \quad .... \quad w_n\}$ and $\sum_{i=1}^{n} w_i$ Where w_i represents the degree of influence of the i-th factor in the lower layer on a certain factor in the upper layer, and the weights of the criterion layer and the decision-making layer are obtained.

3.3 Results

Combining the matrix of the rule layer and the decision layer, we get the weight of the final 14 games as follows：

$$p=\{w_1 \quad w_2 \quad .... \quad w_n\}*w_0$$
$$=\{0.0766\ 0.0749\ 0.0668\ 0.1046\ 0.0699\ 0.0703\ 0.0883\ 0.0652\ 0.0795\ 0.0744\ 0.0764\ 0.0724\ 0.080\}$$

From the results of the analytic hierarchy, it can be known that using P as the evaluation basis, the fourteenth game is the optimal structure in terms of structural strategy and level.

4. Local network analysis model

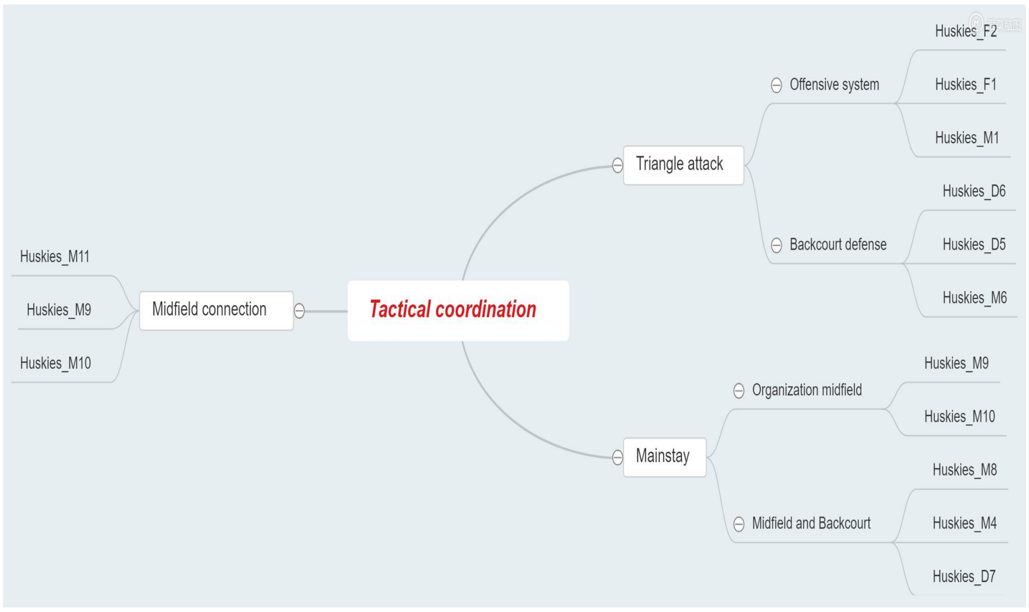
4.1Establishment of the model

From the network performance evaluation model, we analyzed the optimal structure of the game in the 14th game. The local analysis model analyzed the team cooperation from a micro perspective. In the local analysis, the CORNER aggregation subgroup analysis was used[7]. A sub-collection, in which there is a strong, direct and close relationship between the actors in this set. There are two or more actors in this sub-collection. Our analysis through this method can simplify our construction. The complex overall passing network makes it easier for us to find the substructures and their relationships in the network. This allows us to better judge which players are more closely related to each other in order to understand what different groups are in the chosen field of study. Compared with the team, the analyzed group of players has a stronger purpose, mobility and coordination, understand the relationship between each player, and facilitate the coach to arrange the order of players and develop tactics.

4.2 Results

① We use the CORNER agglomerate subgroup analysis method for the constructed passing network to substitute the data of the Husky team game[8], and use the fourteenth field as an example to obtain the following results, which are arranged as follows:

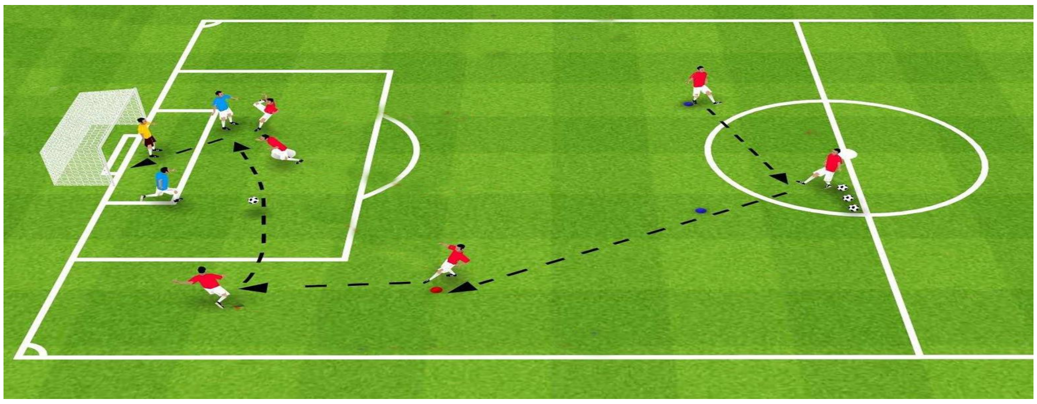
Table 14:Aggregation Subgroup Analysis Team Structure



From the perspective of the micro analysis, we can divide the set of huskies football players into groups with close and positive relationships, such as the two-tuple, three-tuple relationship, analyze the internal structure of the network, and change the cooperation of the football field from Explained from a micro perspective, as shown in the figure, the tactical system can be divided into four main parts: triangle attack, midfield connection, mainstay, and backcourt defense according to the position function.

It can be analyzed by the CORNER aggregation subgroup: it can analyze the corresponding coordination structure and the core members, as shown in the offensive side, are initiated by Huskies_M1. Huskies_F2 and Huskies_F1 are used as a double combination through headers, passes, loose ball duels and offensive shots. The backcourt is guarded by Huskies_D6, Huskies_D5 and Huskies_M6. The air showdown can achieve the goal of defensively blocking the opponent's attack, and pass the stealing ball with a high pass. The pass is simply passed to the midfield player. The midfielder is Huskies_M11, Huskies_M9 and Huskies_M10. To retreat to advance, you can attack and defend. And through the analysis of the 14th game, 4: 0 victory was achieved. Analyzing the "chemical reactions" within the team from a micro perspective makes the team structure strategy more flexible and organized.

Table 15:Illustrated tactical strategy by Photoshop



Some individuals often play a vital role in teamwork. What we need to do is to statistically analyze the individuals who have a strong influence in the network, and judge their specific impact. There are two main aspects: the centrality of the node degree and the maximum value of the feature vector.

②The centrality of the node degree

The centrality of a node degree is to judge the degree of centrality of a node by its out-degree and in-degree. If a node has the highest degree, it is said that the node is located in the center of the local area network and has "power". [4]

To calculate the out-degree of a node, just add each column of the adjacency matrix together to get the column vector **D**

$$D_i = \sum\nolimits_{j=1}^{N} A_{ij}$$

Table 16 the out-degree D in the first game:

Player	First half	Second half	Player	First half	Second half
Huskies_G1	9	3	**Huskies_M3**	21	16
Huskies_F1	13	6	**Huskies_D3**	37	10
Huskies_D1	33	11	**Huskies_D4**	13	10
Huskies_M1	28	23	**Huskies_F3**	5	8
Huskies_F2	21	25	**Huskies_D5**	0	21
Huskies_D2	30	0	**Huskies_M4**	0	9
Huskies_M2	12	3	**Huskies_M5**	0	5

③The centrality of the feature vector

The centrality of the feature vector can reflect the degree of connection between the node and other nodes. Compared with the centrality of the point degree, the centrality of the feature vector integrates the out-degree and in-degree of the node, and can better reflect the degree of individual connection. Knowing where the maximum eigenvalues are, you can know the most important players on the team. In Appendix II, we divide all games into 4 segments, and list the positions of the largest feature values in all time periods. For example, in the first half of the first game, we calculated that the maximum feature vector is 25.8625 and the subscript is 9, indicating that the player Huskies_D3 has the strongest connection with other players, which is consistent with the analysis of the degree of centrality in the previous article.

5 Model validation

5.1 Pearson correlation calculation

In the previous section, it was inaccurate to analyze the relationship between winning and losing and the geometry only from the naked eye in the figure. The following uses mathematical statistics knowledge to calculate the correlation between each index and the game result to verify the accuracy of the model. The first method is to calculate the Pearson linear correlation coefficient, which is used to reflect the linear correlation between two variables X and Y, which can only be used in the case of two-variable correlation analysis.

First of all, quantify the game result, let the game result E=Huskies score-opponent's score. The indicators we want to test are:

- The number of passes
- The average shortest path
- Clustering coefficient
- Offset of the team centroid

Suppose you want to test the correlation between two variables x and y. First calculate the **Pearson correlation coefficient r,** which is the ratio of the covariance of x and y to the standard deviation:

$$r=\frac{\operatorname{cov}(x,y)}{\sigma_x\sigma_y}=\frac{E[(x-\mu_x)(x-\mu_y)]}{\sigma_x\sigma_y}=\frac{\sum_{i=1}^{N}(x-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\bar{x})^2\sum_{i=1}^{N}(y_i-\bar{y})^2}}$$

The correlation between the match result E and each parameter was calculated using matlab, and the correlation coefficient matrix was obtained as follows, and finally the correlation of the variables was verified.

Table 17:Variable correlation verification

lThe number of passes	lThe average shortest path	Clustering coefficient	Offset of the team centroid
>>corrcoef([result,passhus1-passoppo1])	>>corrcoef([result,shorthus1-shortoppo1])	>> corrcoef([result,coeffhus1])	>> corrcoef([result,xhus1])
$R=\begin{pmatrix}1 & 0.3426\\ 0.3426 & 1\end{pmatrix}$	$R=\begin{pmatrix}1 & 0.1811\\ 0.1811 & 1\end{pmatrix}$	$R=\begin{pmatrix}1 & -0.1275\\ -0.1275 & 1\end{pmatrix}$	$R=\begin{pmatrix}1 & -0.1275\\ -0.1275 & 1\end{pmatrix}$

This shows that the Husky team's game score is positively related to the difference between the team's pass number and the opponent's pass number, and the correlation is obvious, which is 0.3426. The final score is also positively related to the difference between the Husky team's average shortest path and the opponent's Less sexual. However, the clustering coefficient and the center-offset distance have little to do with the final score, which is different from what we see with the naked eye.

6. Model evaluation and correction

6.1Advantage

For the overall network analysis, we focused on data mining and processing, using the excellent data-layer language Java to calculate the adjacency matrix from the data given, and combine it with matlab to analyze the data. Finally, we got a large number of feature vector maximums and center positions. Data (Appendix II), the full use of the original data given. Therefore, our model has sufficient data support and credibility.

Social network analysis is more about teamwork, which is undoubtedly very advantageous for studying sports which requires a high degree of cooperation. Although the players continue to move throughout the game, our model explains the importance of passing from the perspective of graph theory.

6.2 Disadvantage

In the geometric statistical analysis of the team, we only considered the position of the centeriod and the dispersion of the team, and did not analyze the specific shape--this is undoubtedly a bit of a talk on paper.

When analyzing the correlation between each parameter and the result of the game, because of we don’t know the specific goal time, can we only analyze it based on the score. This makes the Pearson correlation coefficient seem overwhelming. It is not even as persuasive as observing the bar chart with the naked eye.

Our model analyzes what parameters have a greater impact on the outcome of the game, but all the data is fuzzy, and we are temporarily unable to propose an accurate optimization design for the specific network.

7. Conclusion

7.1 Suggestion on team development

From our analysis, the team's performance has a great relationship with the team's passing network performance. This includes the total number of passes, the aggregation coefficient of the passing network, the robustness of the passing network, and the average shortest path of the network. The position of the team, etc. However, we must optimize the existing match formation based on these data:

First of all, the center of the team must not be too far forward or too late. Pay attention to the attack while stabilizing the defense. In the calculated position data of the team center, the Huskies’ center is almost always slightly behind, and the defense is paid more attention than the opponents. But it is not necessarily that the center moves backwards to hold the ball. Our analysis shows that only by increasing the robustness and clustering coefficient of the passing network and ensuring your team not to get flustered in danger is the best way to defend. If the players are gathered in the backcourt, yet do not understand the cooperation, not only do they can not keep the ball, but lose the opportunity to attack.

In addition, from the perspective of the directed graph, the convenience of passing between players needs to be considered when designing the formation. We call it "average shortest path", which essentially requires the efficiency of the entire network to be passed as much as possible. For offense, we need to have a passing path straight into the front line. It should be flexible enough to prevent opponents from intercepting the ball. For defense, we also need to deploy a tighter passing network in the backcourt to increase the network's robustness.

We also found that teamwork is essential in the game. From the perspective of the micro analysis, we can divide the set of Huskies football players into groups with close and positive relationships, such as the two-tuple, three-tuple relationship. As shown in the figure, the tactical system can be divided into four main parts: triangle attack, midfield connection, mainstay, and backcourt defense.

As shown in the figure, the offensive end was initiated by Huskies_M1, Huskies_F2, Huskies_F1 as a double combination through headers, passes, loose ground duels, and offensive shots. The backcourt is guarded by three players: Huskies_D6, Huskies_D5, and Huskies_M6. To achieve the purpose of blocking the opponent's attack, and pass the stealing ball high, simply pass to the midfield player, and the midfielder Huskies_M11, Huskies_M9, Huskies_M10 are connected to initiate the attack and the simple pass method is the main method. You can attack and defend. And through the analysis of the 14th game, 4: 0 victory was achieved. Analyzing the "chemical reactions" within the team from a micro perspective makes the team structure strategy more flexible and organized.

7.2 Generized model of team performance

For broader teamwork, we must also emphasize the importance of communication between people. A large team is not only composed of individuals, the interconnections between them, and small groups of them are essential. If there is a lack of cooperation between people, rules and constraints, and no consistent goal, then the team will not be able to form, as described in Gustave LeBon's "Psychologies does Fouls".

Through the analysis of the football team in this model, we conclude that a large team consists of small collectives, and in small collectives, people work closely with each other. As is shown in the picture below, small groups, bound by the same rules, having the same goals and even beliefs, cooperate with each other and then form a larger group.

7.3 Extended application of the modal and Further Discussion

In our social network analysis model, we analyze football cooperation strategies from local and global perspectives. If we introduce the requirements of other fields, we can also extend the model method to analyze the network, achieve more social cooperation levels such as the aggregation of expert teams, and then effectively analyze different cooperation strategies, which shows that this model is very useful Forward-looking.

Our model is combined with relevant knowledge and is in line with mathematical principles. It can combine different parameters to improve and achieve more accurate analysis of cooperation strategies. It can play a greater role in cooperation in today's challenging era and make cooperation a successful magic weapon.

8.References

[1] Jiang Xin. Application Research of Social Network Analysis Method in Library and Information Science [M]. Beijing: Intellectual Property Press, 2015, 6: 1.

[2] Yufan Zheng. Graph analysis-clustering coefficient https://alphafan.github.io/posts/graph_analysis.html

[3] Lin Yue. Research on Optimal Design of Tree Structure Stability in Graph Theory Science Bulletin 2016.2

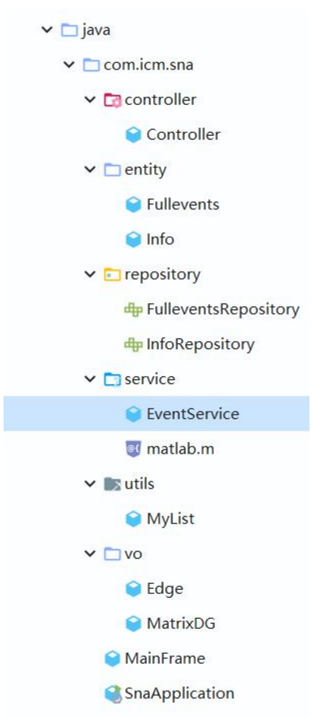
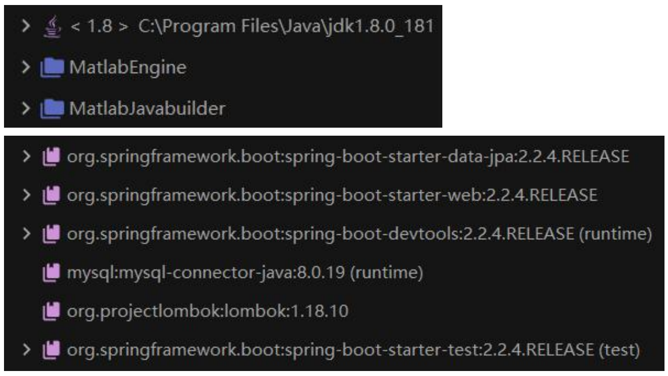
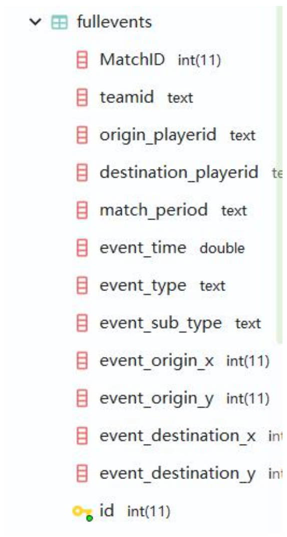
[4] Li Bo, Wang Lei. Feasibility Analysis of Social Network Analysis on Passing Performance in Football Matches. Journal of Beijing Sport University 2017.8

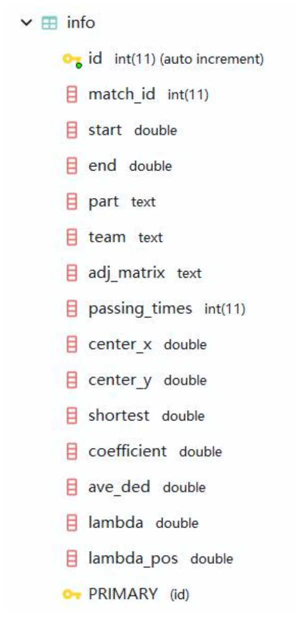
[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6753100/#CR30
[6]Zhang Yan. Application and Research Progress of Social Network Analysis in Football Performance [C]. Chinese Sports Science Society. Compilation of Abstracts of the Eleventh National Sports Science Conference. Chinese Sports Science Society: Chinese Sports Science Society,
[7] Sheng Kerong, Yang Yu, Zhang Hongxia. Study on the agglomeration subgroups and influencing factors of Chinese urban networks [J]. Geographical Research, 2019, 38 (11): 2639-2652.
[8]Wan Huaiyu. Research on agglomerative subgroup analysis of large-scale interaction data sets [D]. Beijing Jiaotong University, 2007

9.Appendix

Appendix1:JAVA Program:

Our Java spring data jpa program connects matlab and mysql database to process data from the table fullevents to the table info. The table fullevents is imported from the file fullevents.csv. The output is to the table info,which has been dumped into the file Appendix2.

项目结构: **project structure**	**项目依赖，**包含 maven 与本地： project dependency,incluing maven and local.	**数据表结构：** tables structure.
 java com.icm.sna controller Controller entity Fullevents Info repository FulleventsRepository InfoRepository service EventService matlab.m utils MyList vo Edge MatrixDG MainFrame SnaApplication	 < 1.8 > C:\Program Files\Java\jdk1.8.0_181 MatlabEngine MatlabJavabuilder org.springframework.boot:spring-boot-starter-data-jpa:2.2.4.RELEASE org.springframework.boot:spring-boot-starter-web:2.2.4.RELEASE org.springframework.boot:spring-boot-devtools:2.2.4.RELEASE (runtime) mysql:mysql-connector-java:8.0.19 (runtime) org.projectlombok:lombok:1.18.10 org.springframework.boot:spring-boot-starter-test:2.2.4.RELEASE (test)	 fullevents MatchID int(11) teamid text origin_playerid text destination_playerid te match_period text event_time double event_type text event_sub_type text event_origin_x int(11) event_origin_y int(11) event_destination_x in event_destination_y in id int(11)

```
info
  id  int(11) (auto increment)
  match_id  int(11)
  start  double
  end  double
  part  text
  team  text
  adj_matrix  text
  passing_times  int(11)
  center_x  double
  center_y  double
  shortest  double
  coefficient  double
  ave_ded  double
  lambda  double
  lambda_pos  double
  PRIMARY  (id)
```

SnaApplication.java

```
package com.icm.sna;

import com.mathworks.engine.EngineException;
import com.mathworks.engine.MatlabEngine;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

/**
 * 本程序的启动类 the start class
 */
@SpringBootApplication
public class SnaApplication {
    /**
     * 将 MatlabEngine 交由 spring 管理 hand over the Matlab Engine to spring to manage
     */
    @Bean
    public MatlabEngine startMatlab() {
        try {
            return MatlabEngine.startMatlab();
        } catch (EngineException | InterruptedException e) {
            e.printStackTrace();
        }
        return null;
    }

    public static void main(String[] args) {
        SpringApplication.run(SnaApplication.class, args);
    }

}
```

EventService.java

```
package com.icm.sna.service;

import com.icm.sna.repository.FulleventsRepository;
import com.icm.sna.vo.Edge;
import com.icm.sna.vo.MatrixDG;
import com.mathworks.engine.MatlabEngine;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutionException;

/**
 * 本程序的核心业务逻辑 the center service logic of this program
 */
@Service
public class EventService {

    @Autowired
    private FulleventsRepository fulleventsRepository;

    @Autowired
    private MatlabEngine engine;

    @Autowired
    private InfoRepository infoRepository;

    public void getPassingEvent(int matchId, double time, double stays, String part) {
        System.out.println("Match " + matchId + " " + part + " " + time + " " + stays);
        Info info = new Info();
        Info info1 = new Info();
        info.setTeam("Huskies");
        info.setStart(time);
        info.setEnd(time + stays);
        info.setMatchId(matchId);
        info.setPart(part);
        info1.setTeam("Opponents");
        info1.setStart(time);
        info1.setEnd(time + stays);
        info1.setMatchId(matchId);
        info1.setPart(part);
        //查询该比赛中有哪些队员，并分成两队 to query how many player in this match, divide in
to two
        List<String> huskies = new ArrayList<>();
        List<String> oppos = new ArrayList<>();
        fulleventsRepository.findDistinctByOriginPlayerId(matchId, time, time + stays,
part).forEach(s -> {
            if (s != null && !s.isEmpty()) {
                if (s.contains("Huskies")) huskies.add(s);
                if (s.contains("Opponent")) oppos.add(s);
            }
        });
        fulleventsRepository.findDistinctByDesPlayerId(matchId, time, time + stays,
part).forEach(s -> {
            if (s != null && !s.isEmpty()) {
                if (s.contains("Huskies")) if (!huskies.contains(s)) huskies.add(s);
                if (s.contains("Opponent")) if (!oppos.contains(s)) oppos.add(s);
            }
        });
        //遍历这些队员，分别输出他们的平均位置 foreach these players and output their average
locations
        System.out.println("huskies: " + huskies);
        Locations husLoc = printLoc(matchId, huskies, time, time + stays, part);
        System.out.println("opponent: " + oppos);
        Locations oppoLoc = printLoc(matchId, oppos, time, time + stays, part);

        //查询传球记录 query for pass events
        List<Edge> edgesHus = new ArrayList<>();
        List<Edge> edgesOppo = new ArrayList<>();
//        fulleventsRepository.findDistinctByPasses(matchId, "Pass").forEach(edge -> {
//            if (edge.getDestination() != null) {
//                                fulleventsRepository.findPasses(edge.getOrigin(),
edge.getDestination()).forEach(fullevents -> {
//                                if (fullevents.getTeamId().contains("Huskies"))
edgesHus.add(edge);
//                               if (fullevents.getTeamId().contains("Opponent"))
edgesOppo.add(edge);
//                });
//            }
//        });
        fulleventsRepository.findByMatchId(matchId, time, time + stays,
part).forEach(fullevents -> {
```

```
            if        (fullevents.getOriginPlayerId()        !=        null        &&
fullevents.getDestinationPlayerId() != null) {
                if (fullevents.getTeamId().contains("Huskies"))
                    edgesHus.add(new              Edge(fullevents.getOriginPlayerId(),
fullevents.getDestinationPlayerId()));
                if (fullevents.getTeamId().contains("Opponent"))
                    edgesOppo.add(new             Edge(fullevents.getOriginPlayerId(),
fullevents.getDestinationPlayerId()));
//                                                                                 if
(fullevents.getOriginPlayerId().equals(fullevents.getDestinationPlayerId()))
//                    System.err.println(fullevents.getId());
            }
        });

        System.out.println("Huskies' passing times:" + edgesHus.size());
        info.setPassingTimes(edgesHus.size());
        System.out.println("Opponent's passing times:" + edgesOppo.size());
        info1.setPassingTimes(edgesOppo.size());
        //生成邻接矩阵并输出 output adj matrix
        System.out.println("Huskies matrix in the match " + matchId);
        String matH = new MatrixDG(huskies, edgesHus).print();
        info.setAdjMatrix(matH);

        System.out.println("Opponents matrix in the match " + matchId);
        String matO = new MatrixDG(oppos, edgesOppo).print();
        info1.setAdjMatrix(matO);

        //启动 matlab 引擎 start matlab
        System.out.println("Matlab starting...");
        try {
            engine.eval("matH=" + matH);//哈士奇队的邻接矩阵
            engine.eval("matO=" + matO);//对面的邻接矩阵
            engine.eval("X=" + husLoc.avex + ";");//哈士奇队球员 X 坐标向量
            engine.eval("Y=" + husLoc.avey + ";");
            engine.eval("X2=" + oppoLoc.avex + ";");
            engine.eval("Y2=" + oppoLoc.avey + ";");

            System.out.println("点度中心度");
            engine.eval("sum(matH)");

            System.out.println("哈士奇队伍中心点位置");
            engine.eval("siz=size(X);\n");
            engine.eval("cenx=sum(X)/siz(2)\n");
            info.setCenterX(engine.getVariable("cenx"));
```

```
engine.eval("ceny=sum(Y)/siz(2)\n");
info.setCenterY(engine.getVariable("ceny"));
engine.eval("density=mean(((X-cenx).^2+(Y-ceny).^2).^0.5)");
info.setDesity(engine.getVariable("density"));
System.out.println("对手队伍中心点位置");
engine.eval("siz=size(X2);\n");
engine.eval("cenx=sum(X2)/siz(2)\n");
info1.setCenterX(engine.getVariable("cenx"));
engine.eval("ceny=sum(Y2)/siz(2)\n");
info1.setCenterY(engine.getVariable("ceny"));
engine.eval("density=mean(((X2-cenx).^2+(Y2-ceny).^2).^0.5)");
info1.setDesity(engine.getVariable("density"));

System.out.println("哈士奇特征向量与特征值矩阵");
engine.eval("[V,D]=eig(matH)");//求特征向量与特征值矩阵
System.out.println("特征向量最大值和位置");
engine.eval("[Lambda,i]=max(diag(D))");//特征向量最大值和位置
info.setLambda(engine.getVariable("Lambda"));
info.setLambdaPos(engine.getVariable("i"));
System.out.println("对手特征向量与特征值矩阵");
engine.eval("[V,D]=eig(matO);");//求特征向量与特征值矩阵
System.out.println("特征向量最大值和位置");
engine.eval("[Lambda,i]=max(diag(D))");//特征向量最大值和位置
info1.setLambda(engine.getVariable("Lambda"));
info1.setLambdaPos(engine.getVariable("i"));

System.out.println("求平均最短路径");
engine.eval("aa=[];\n" +
        "    siz=size(matH);\n" +
        "    siz=siz(1);\n" +
        "    for k = 1:siz\n" +
        "        aa=[aa;graphshortestpath(sparse(matH),k)];\n" +
        "    end\n" +
        "    s=sum(sum(aa))/(siz*(siz-1));");
Double s = engine.getVariable("s");
info.setShortest(s);
engine.eval("aa=[];\n" +
        "    siz=size(matO);\n" +
        "    siz=siz(1);\n" +
        "    for k = 1:siz\n" +
        "        aa=[aa;graphshortestpath(sparse(matO),k)];\n" +
        "    end\n" +
        "    s=sum(sum(aa))/(siz*(siz-1));");
info1.setShortest(engine.getVariable("s"));
```

```
            System.out.println("计算集聚系数");
            info.setCoefficient(coefficient("matH"));
            info1.setCoefficient(coefficient("matO"));
            System.out.println("计算度分布");
            info.setAveDed(degree("matH"));
            info1.setAveDed(degree("matO"));
            info.checkInf();
            info1.checkInf();
            infoRepository.save(info);
            infoRepository.save(info1);
//            getPassingEvent2(matchId, time, stays);
//            drawNodes();
            drawDigraph();
        } catch (InterruptedException | ExecutionException e) {
            e.printStackTrace();
        }
    }

    /**
     * DeD————————网络图各节点的度分布
     * aver_DeD———————网络图的平均度
     *
     * @param s 邻接矩阵名称
     */
    private Double degree(String s) throws ExecutionException, InterruptedException {
        engine.eval("N=size(" + s + ",2);\n" +
                "DeD=zeros(1,N)\n" +
                "for i=1:N\n" +
                "   DeD(i)=sum(" + s + "(i,:));\n" +
                "end\n" +
                "aver_DeD=mean(DeD)");
        return engine.getVariable("aver_DeD");
    }

    /**
     * 使用 matlab 计算集聚系数
     *
     * @param s 邻接矩阵在 matlab 中的名称
     */
    private    Double    coefficient(String    s)    throws    ExecutionException,
InterruptedException {
        engine.eval("trace=" + s +
                "\n N = size(trace,1); % number of nodes\n" +
                "clustering_nodes = zeros(N,1);\n" +
                "for i = 1:N\n" +
                "    t_transitive = [];\n" +
                "    for t1 = 1:size(trace,3)\n" +
                "        neighbors = setdiff(find(trace(i,:,t1)==1),i);\n" +
                "        if ~isempty(neighbors)\n" +
                "            for k = 1:length(neighbors)\n" +
                "                for t2 = t1:size(trace,3)\n" +
                "                                                  neighbors_of_neighbor   =
setdiff(find(trace(neighbors(k),:,t2)==1),[neighbors(k) i]);\n" +
                "                    if ~isempty(neighbors_of_neighbor)\n" +
                "                        for l = 1:length(neighbors_of_neighbor)\n" +
                "                                                                        t3   =
find(trace(neighbors_of_neighbor(l),i,t2:end)==1,1);\n" +
                "                            if ~isempty(t3)\n" +
                "                                t_transitive = [t_transitive t3];\n" +
                "                            end\n" +
                "                        end\n" +
                "                        break;\n" +
                "                    end\n" +
                "                end\n" +
                "            end\n" +
                "            break;\n" +
                "        end\n" +
                "    end\n" +
                "   \n" +
                "    if isempty(t_transitive)\n" +
                "        clustering_nodes(i) = 0;\n" +
                "    else\n" +
                "        clustering_nodes(i) = 1/min(t_transitive);\n" +
                "    end\n" +
                "end\n" +
                "c = mean(clustering_nodes)\n");
        return engine.getVariable("c");
    }

    private void drawNodes() throws ExecutionException, InterruptedException {
        engine.eval("scatter(X,Y,[],[0,1,0],'filled');\n" +
                "hold on;\n" +
                "scatter(X2,Y2,[],[1,0,0],'filled');\n" +
                "siz=size(X);\n" +
                "sum(X)/siz(2)" +
                "sum(Y)/siz(2)" +
                "scatter(sum(X)/siz(2),sum(Y)/siz(2),[],[0,0.5,0],'filled');\n" +
                "siz=size(X2);\n" +
                "scatter(sum(X2)/siz(2),sum(Y2)/siz(2),[],[0.5,0,0],'filled');\n" +
                "%" +

"a1=full(adjacency(graph(int8(X),int8(Y),(X.^2+Y.^2).^0.5),'weighted'));\n" +
                "%" +

"a2=full(adjacency(graph(int8(X2),int8(Y2),(X2.^2+Y2.^2).^0.5),'weighted'));\n" +
                "    " +
                "siz=size(X);\n" +
                "    result=[];\n" +
                "    for ii=[1:siz(2)]\n" +
                "        for jj=[1:siz(2)]\n" +
                "            result(ii,jj)=(X(ii).^2+Y(jj).^2).^0.5;\n" +
                "        end\n" +
                "    end\n" +
                "    a1=result\n" +
                "    " +
                "siz=size(X2);\n" +
                "    result=[];\n" +
                "    for ii=[1:siz(2)]\n" +
                "        for jj=[1:siz(2)]\n" +
                "            result(ii,jj)=(X2(ii).^2+Y2(jj).^2).^0.5;\n" +
                "        end\n" +
                "    end\n" +
                "    a2=result"
        );
        coefficient("a1");
        coefficient("a2");
    }

    private void drawDigraph() throws ExecutionException, InterruptedException {
        //画有向图
        engine.eval("G=digraph(matH);");
        engine.eval("G2=digraph(matO);");
        engine.eval("p=plot(gca,G,'XData',X,'YData',Y)");
        engine.eval("hold on");
        engine.eval("p2=plot(gca,G2,'XData',X2,'YData',Y2)");
        //设置线宽粗细
        engine.eval("G.Nodes.NodeColors = outdegree(G);\n" +
                "p.NodeCData = G.Nodes.NodeColors;\n" +
                "G2.Nodes.NodeColors = outdegree(G2);\n" +
                "p2.NodeCData = G2.Nodes.NodeColors;\n" +
                "colorbar");
        engine.eval("p.LineWidth = 7*G.Edges.Weight/max(G.Edges.Weight);");
        engine.eval("p2.LineWidth = 7*G2.Edges.Weight/max(G2.Edges.Weight);");
    }

    /**
     * 研究特定时间段内传球矢量图
     */
    public void getPassingEvent2(int matchId, double time, double stays) throws
ExecutionException, InterruptedException {
        List<Double> edgesHusox = fulleventsRepository.findByTimeox(matchId, time, time
+ stays, "Huskies");
        List<Double> edgesHusoy = fulleventsRepository.findByTimeoy(matchId, time, time
+ stays, "Huskies");
        List<Double> edgesOppox = fulleventsRepository.findByTimeox(matchId, time, time
+ stays, "Opponent");
        List<Double> edgesOppoy = fulleventsRepository.findByTimeoy(matchId, time, time
+ stays, "Opponent");
        List<Double> edgesHusdx = fulleventsRepository.findByTimedx(matchId, time, time
+ stays, "Huskies");
        List<Double> edgesHusdy = fulleventsRepository.findByTimedy(matchId, time, time
+ stays, "Huskies");
        List<Double>  edgesOppodx  =  fulleventsRepository.findByTimedx(matchId,  time,
time + stays, "Opponent");
        List<Double>  edgesOppody  =  fulleventsRepository.findByTimedy(matchId,  time,
time + stays, "Opponent");
        handleArray(edgesHusox, edgesHusdx);
        handleArray(edgesHusoy, edgesHusdy);
        handleArray(edgesOppox, edgesOppodx);
        handleArray(edgesOppoy, edgesOppody);
        engine.eval("quiver(" + edgesHusox + "," + edgesHusoy + "," + edgesHusdx + ","
+ edgesHusdy + ")");
        engine.eval("hold on");
        engine.eval("quiver(" + edgesOppox + "," + edgesOppoy + "," + edgesOppodx + ","
+ edgesOppody + ")");
    }

    /**
     * 清除查询坐标为 null 的元素 a
     */
    private void handleArray(List<Double> a, List<Double> b) {
        for (int i = 0; i < a.size(); i++) {
            if (a.get(i) == null) {
                if (b.get(i) != null)
                    a.set(i, b.get(i));
                else a.set(i, 0.0);
            }
        }
        for (int i = 0; i < b.size(); i++) {
            if (b.get(i) == null) {
                if (a.get(i) != null)
                    b.set(i, a.get(i));
                else b.set(i, 0.0);
            }
        }
    }

    /**
     * 输出这些球员本场比赛的平均位置
     */
    private Locations printLoc(int matchId, List<String> players, double time, double
stays, String part) {
        ArrayList<Double> avex = new ArrayList<>();
        ArrayList<Double> avey = new ArrayList<>();

        for (String s : players) {
            Long sum_x = fulleventsRepository.sum_x(matchId, s, time, stays, part);
            Long sum_y = fulleventsRepository.sum_y(matchId, s, time, stays, part);
            if (sum_x != null && sum_y != null) {
                avex.add((double) sum_x /
                        fulleventsRepository.count_x(matchId, s, time, stays, part));
                avey.add((double) sum_y /
                        fulleventsRepository.count_x(matchId, s, time, stays, part));
            }
        }
        System.out.println("average X location " + avex);
        System.out.println("average Y location " + avey);
        return new Locations(avex, avey);
    }

    static class Locations {
        ArrayList<Double> avex;
        ArrayList<Double> avey;

        public Locations(ArrayList<Double> avex, ArrayList<Double> avey) {
            this.avex = avex;
            this.avey = avey;
        }
    }
}
```

Edge.java
```
package com.icm.sna.vo;

import lombok.Data;

/**
 * 有向图的边
 */
@Data
public class Edge {
    private String origin;
    private String destination;

    public Edge(String origin, String destination) {
        this.origin = origin;
        this.destination = destination;
    }
}
```
MatrixDG.java
```
package com.icm.sna.vo;

import java.util.Arrays;
import java.util.List;

/**
 * 邻接矩阵有向图
 */
public class MatrixDG {
    int size;
    List<String> vertexs;
    int[][] matrix;

    public MatrixDG(List<String> vertexs, List<Edge> edges) {
        size = vertexs.size();
        matrix = new int[size][size];
        for (int[] ints : matrix) Arrays.fill(ints, 0);//构造 0 矩阵
        this.vertexs = vertexs;
        //和邻接矩阵无向图差别仅仅在这里
        for (Edge edge : edges) {
//            System.out.println(edge.getOrigin() + " " + edge.getDestination());
            int p1 = findVertex(edge.getOrigin());
            int p2 = findVertex(edge.getDestination());
            if (p1 == -1 || p2 == -1) continue;
            matrix[p1][p2]++;
        }
    }

    public String print() {
        int sum = 0;
        StringBuilder stringBuilder = new StringBuilder();
        stringBuilder.append('[');
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                sum += matrix[i][j];
                if (j == matrix[i].length - 1) {
                    stringBuilder.append(matrix[i][j]);
                    break;
                }
                stringBuilder.append(matrix[i][j]).append(",");
            }
            if (i == matrix.length - 1) break;
            stringBuilder.append(";");
        }
        stringBuilder.append("]");
        System.out.println(stringBuilder);
        System.out.println("sum=" + sum);
        return new String(stringBuilder);
    }

    private int findVertex(String s) {
        int a = 0;
        for (String ss : vertexs) {
            if (ss.equals(s)) return a;
            a++;
        }
        System.out.println("unfound:" + s);
        return -1;
    }
}
```

FulleventsRepository.java

```
package com.icm.sna.repository;

import com.icm.sna.entity.Fullevents;
import com.icm.sna.vo.Edge;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.util.ArrayList;
import java.util.List;

/**
 * jpa 数据库访问接口，执行 HQL 语言
 */
@Repository
public interface FulleventsRepository extends JpaRepository<Fullevents, Integer> {
    @Query("select e from Fullevents e where e.matchId=?1 and e.eventTime>?2 and
e.eventTime<?3 and e.matchPeriod like ?4")
    List<Fullevents> findByMatchId(int matchID, double start, double end, String part);

    @Query("select e.eventDestinationX-e.eventOriginX from Fullevents e where
e.matchId=?1" +
            " and e.eventTime>?2 and e.eventTime<?3 and e.teamId like concat('%',?4,'%')
and e.eventType not like '%Substitution%' and e.eventType not like 'Foul'")
    List<Double> findByTimedx(int matchID, double start, double end, String team);

    @Query("select e.eventDestinationY-e.eventOriginY from Fullevents e where
e.matchId=?1 " +
            "and e.eventTime>?2 and e.eventTime<?3 and e.teamId like concat('%',?4,'%')
and e.eventType not like '%Substitution%' and e.eventType not like 'Foul'")
    List<Double> findByTimedy(int matchID, double start, double end, String team);

    @Query("select e.eventOriginX from Fullevents e where e.matchId=?1 and
e.eventTime>?2 " +
            "and e.eventTime<?3 and e.teamId like concat('%',?4,'%') and e.eventType not
like '%Substitution%' and e.eventType not like 'Foul'")
    List<Double> findByTimeox(int matchID, double start, double end, String team);

    @Query("select e.eventOriginY from Fullevents e where e.matchId=?1 and
e.eventTime>?2 " +
            "and e.eventTime<?3 and e.teamId like concat('%',?4,'%') and e.eventType not
like '%Substitution%' and e.eventType not like 'Foul'")
    List<Double> findByTimeoy(int matchID, double start, double end, String team);

    //      @Query("select distinct e.originPlayerId from Fullevents e where
e.matchId=?1")
    @Query("select distinct e.originPlayerId from Fullevents e where e.matchId =?1 and
e.eventTime>?2 and e.eventTime<?3 and e.matchPeriod like ?4")
    List<String> findDistinctByOriginPlayerId(int matchID, double start, double end,
String part);

    @Query("select distinct e.destinationPlayerId from Fullevents e where e.matchId=?1
and e.eventTime>?2 and e.eventTime<?3 and e.matchPeriod like ?4")
    List<String> findDistinctByDesPlayerId(int matchID, double start, double end, String part);

    @Query("select e from Fullevents e where e.matchId=?1 and e.eventType=?2")
    List<Fullevents> findByEventType(int matchId, String eventType);

    @Query("select e from Fullevents e where e.destinationPlayerId=?2 and e.originPlayerId=?1")
    List<Fullevents> findPasses(String origin, String dest);

    @Query("select distinct new com.icm.sna.vo.Edge(e.originPlayerId, e.destinationPlayerId) from Fullevents e where e.matchId=?1 and e.eventType=?2")
    List<Edge> findDistinctByPasses(int matchId, String type);

    @Query("select sum(f.eventDestinationX) from Fullevents f where f.matchId=?1 and f.originPlayerId=?2 and f.eventTime>?3 and f.eventTime<?4 and f.matchPeriod like ?5")
    Long sum_x(int matchId, String player, double start, double end, String part);

    @Query("select sum(f.eventDestinationY) from Fullevents f where f.matchId=?1 and f.originPlayerId=?2 and f.eventTime>?3 and f.eventTime<?4 and f.matchPeriod like ?5")
    Long sum_y(int matchId, String player, double start, double end, String part);

    @Query("select count(f) from Fullevents f where f.matchId=?1 and f.originPlayerId=?2 and f.eventTime>?3 and f.eventTime<?4 and f.matchPeriod like ?5")
    Long count_x(int matchId, String player, double start, double end, String part);}
```

Controller.java

```
import com.icm.sna.service.EventService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

/**
 * 由于一些 bug，service 层的单元测试未能成功运行，于是写了一个控制器，使用浏览器访问运行本程序。
 */
@RestController
public class Controller {
    @Autowired
    private EventService eventService;
    @RequestMapping(value = "/start",method = RequestMethod.GET)
    public void starter(int id){
        eventService.getPassingEvent(id);
    }
}
```

Fullevents.java

```
package com.icm.sna.entity;

import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import java.util.Objects;

/**
 * 用于数据访问的实体类
 */
@Entity
public class Fullevents {
    private Integer matchId;
    private String teamId;
    private String originPlayerId;
    private String destinationPlayerId;
    private String matchPeriod;
    private String eventTime;
    private String eventType;
    private String eventSubType;
    private Integer eventOriginX;
    private Integer eventOriginY;
    private Integer eventDestinationX;
    private Integer eventDestinationY;
    private int id;

    @Basic
    @Column(name = "MatchID")
    public Integer getMatchId() {
        return matchId;
    }

    public void setMatchId(Integer matchId) {
        this.matchId = matchId;
    }

    @Basic
    @Column(name = "teamid")
    public String getTeamId() {
        return teamId;
    }

    public void setTeamId(String teamId) {
        this.teamId = teamId;
    }

    @Basic
    @Column(name = "origin_playerid")
    public String getOriginPlayerId() {
        return originPlayerId;
    }

    public void setOriginPlayerId(String originPlayerId) {
        this.originPlayerId = originPlayerId;
    }

    @Basic
    @Column(name = "destination_playerid")
    public String getDestinationPlayerId() {
        return destinationPlayerId;
    }

    public void setDestinationPlayerId(String destinationPlayerId) {
        this.destinationPlayerId = destinationPlayerId;
    }

    @Basic
    @Column(name = "match_period")
    public String getMatchPeriod() {
        return matchPeriod;
    }

    public void setMatchPeriod(String matchPeriod) {
        this.matchPeriod = matchPeriod;
    }

    @Basic
    @Column(name = "event_time")
    public String getEventTime() {
        return eventTime;
    }

    public void setEventTime(String eventTime) {
        this.eventTime = eventTime;
    }

    @Basic
    @Column(name = "event_type")
    public String getEventType() {
        return eventType;
    }

    public void setEventType(String eventType) {
        this.eventType = eventType;
    }

    @Basic
    @Column(name = "event_sub_type")
    public String getEventSubType() {
        return eventSubType;
    }

    public void setEventSubType(String eventSubType) {
        this.eventSubType = eventSubType;
    }

    @Basic
    @Column(name = "event_origin_x")
    public Integer getEventOriginX() {
        return eventOriginX;
    }

    public void setEventOriginX(Integer eventOriginX) {
        this.eventOriginX = eventOriginX;
    }

    @Basic
    @Column(name = "event_origin_y")
    public Integer getEventOriginY() {
        return eventOriginY;
    }

    public void setEventOriginY(Integer eventOriginY) {
        this.eventOriginY = eventOriginY;
    }

    @Basic
    @Column(name = "event_destination_x")
    public Integer getEventDestinationX() {
        return eventDestinationX;
    }

    public void setEventDestinationX(Integer eventDestinationX) {
        this.eventDestinationX = eventDestinationX;
    }

    @Basic
    @Column(name = "event_destination_y")
    public Integer getEventDestinationY() {
        return eventDestinationY;
    }

    public void setEventDestinationY(Integer eventDestinationY) {
        this.eventDestinationY = eventDestinationY;
    }

    @Id
    @Column(name = "ID")
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

Info.java

```
package com.icm.sna.entity;

import lombok.Data;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import static java.lang.Double.POSITIVE_INFINITY;

@Entity
@Data
public class Info {
    @Id
    @GeneratedValue
    @Column(name="id")
    private Integer id;
    private Integer matchId;
    private Double start;
    private Double end;
    private String part;
    private String team;
    private String adjMatrix;
    private Integer passingTimes;
    private Double centerX;
    private Double centerY;
    private Double shortest;
    private Double coefficient;
    private Double aveDed;
    private Double lambda;
    private Double lambdaPos;
    private Double desity;

    public void checkInf(){
        if(Double.isInfinite(shortest))shortest=-1d;
        if(Double.isInfinite(coefficient))coefficient=-1d;
        if(Double.isInfinite(aveDed))aveDed=-1d;
        if(Double.isInfinite(lambda))lambda=-1d;
        if(Double.isInfinite(lambdaPos))lambdaPos=-1d;
        if(Double.isInfinite(desity))desity=-1d;
    }
}
InfoRepository.java
package com.icm.sna.repository;

import com.icm.sna.entity.Info;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface InfoRepository extends JpaRepository<Info,Integer> {

}
```

Appendix1:JAVA Program:

MATLAB Codes:

```
function ahpactor
%指标包括：比赛获得分数，比赛失去分数，聚类系数（稳定性），最大特征值
```

```
（灵活性)，传球数
%比赛取定获胜比赛
A = [1/1    2/1    3/2    2/1 3/2
        1/2    1/1    3/4    1/1 3/4
        2/3    4/3    1/1    4/3 1/1
        1/2    1/1    3/4    1/1 3/4
        2/3    4/3    1/1    4/3 1/1];
[w, CR] = AHP(A);
%比赛号码：1 6 11 14 15 17 18 25 27 30 31 35 36
% 比赛获得分数
A1 = [1/1    1/2    1/3 1/4 1/2 1/1 1/3 1/2 1/3 1/2 1/2 1/1 1/2
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1
         3/1    3/2    1/1 3/4 3/2 3/1 1/1 3/2 1/1 3/2 3/2 3/1 3/2
         4/1    2/1    4/3 1/1 2/1 4/1 4/3 2/1 4/3 2/1 2/1 4/1 2/1
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1
         1/1    1/2    1/3 1/4 1/2 1/1 1/3 1/2 1/3 1/2 1/2 1/1 1/2
         3/1    3/2    1/1 3/4 3/2 3/1 1/1 3/2 1/1 3/2 3/2 3/1 3/2
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1
         3/1    3/2    1/1 3/4 3/2 3/1 1/1 3/2 1/1 3/2 3/2 3/1 3/2
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1
         1/1    1/2    1/3 1/4 1/2 1/1 1/3 1/2 1/3 1/2 1/2 1/1 1/2
         2/1    1/1    2/3 1/2 1/1 2/1 2/3 1/1 2/3 1/1 1/1 2/1 1/1];
[w1, CR1] = AHP(A1);

% 比赛丢球    0-3 1-2 2-1
%比赛号码：1 6 11 14 15 17 18 25 27 30 31 35 36
A2 = [1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         2/3    1/1    2/1 2/3 2/3 2/3 1/1 1/1 1/1 2/3 1/1 2/3 2/3
         1/3    1/2    1/1 1/3 1/3 1/3 1/2 1/2 1/2 1/3 1/2 1/3 1/3
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         2/3    1/1    2/1 2/3 2/3 2/3 1/1 1/1 1/1 2/3 1/1 2/3 2/3
         2/3    1/1    2/1 2/3 2/3 2/3 1/1 1/1 1/1 2/3 1/1 2/3 2/3
         2/3    1/1    2/1 2/3 2/3 2/3 1/1 1/1 1/1 2/3 1/1 2/3 2/3
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         2/3    1/1    2/1 2/3 2/3 2/3 1/1 1/1 1/1 2/3 1/1 2/3 2/3
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1
         1/1    3/2    3/1 1/1 1/1 1/1 3/2 3/2 3/2 1/1 3/2 1/1 1/1];
[w2, CR2] = AHP(A2);

% 聚类系数    0-3 1-2 2-1
%比赛号码：1 6 11 14 15 17 18 25 27 30 31 35 36
```

```
A3 = [1/1    7/6    7/6 7/6 7/6 7/5 1/1 7/5 7/5 7/4 7/6 7/6 1/1
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         5/7    5/6    5/6 5/6 5/6 1/1 5/7 1/1 1/1 5/4 5/6 5/6 5/7
         1/1    7/6    7/6 7/6 7/6 7/5 1/1 7/5 7/5 7/4 7/6 7/6 1/1
         5/7    5/6    5/6 5/6 5/6 1/1 5/7 1/1 1/1 5/4 5/6 5/6 5/7
         5/7    5/6    5/6 5/6 5/6 1/1 5/7 1/1 1/1 5/4 5/6 5/6 5/7
         4/7    4/6    4/6 4/6 4/6 5/6 4/7 4/5 4/5 1/1 4/6 4/6 4/7
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         6/7    1/1    1/1 1/1 1/1 6/5 6/7 6/5 6/5 6/4 1/1 1/1 6/7
         1/1    7/6    7/6 7/6 7/6 7/5 1/1 7/5 7/5 7/4 7/6 7/6 1/1];
[w3, CR3] = AHP(A3);

%最大特征值（灵活性)
%比赛号码：1 6 11 14 15 17 18 25 27 30 31 35 36
A4 = [1/1    10/8 10/4 10/9 10/7 10/8 10/8 10/6 10/7 10/8 10/9 10/9 10/7
         8/10    1/1    2/1 8/9 8/7 1/1 1/1 8/6 8/7 1/1 8/9 8/9 8/7
         4/10    1/2    1/1 4/9 4/7 1/2 1/2 4/6 4/7 1/2 4/9 4/9 4/7
         9/10    9/8    9/4 1/1 9/7 9/8 9/8 9/6 9/7 9/8 1/1 1/1 9/7
         7/10    7/8    7/4 7/9 1/1 7/8 7/8 7/6 1/1 7/8 7/9 7/9 1/1
         8/10    1/1    2/1 8/9 8/7 1/1 1/1 8/6 8/7 1/1 8/9 8/9 8/7
         8/10    1/1    2/1 8/9 8/7 1/1 1/1 8/6 8/7 1/1 8/9 8/9 8/7
         6/10    6/8    6/4 6/9 6/7 6/8 6/8 1/1 6/7 6/8 6/9 6/9 6/7
         7/10    7/8    7/4 7/9 1/1 7/8 7/8 7/6 1/1 7/8 7/9 7/9 1/1
         8/10    1/1    2/1 8/9 8/7 1/1 1/1 8/6 8/7 1/1 8/9 8/9 8/7
         9/10    9/8    9/4 1/1 9/7 9/8 9/8 9/6 9/7 9/8 1/1 1/1 9/7
         9/10    9/8    9/4 1/1 9/7 9/8 9/8 9/6 9/7 9/8 1/1 1/1 9/7
         7/10    7/8    7/4 7/9 1/1 7/8 7/8 7/6 1/1 7/8 7/9 7/9 1/1];
[w4, CR4] = AHP(A4);
%平均最短路径
A5= [1/1    9/8    9/4 1/1 9/4 9/10 9/8 9/6 9/7 9/8 9/8 1/1 9/8
       8/9    1/1    2/1 8/9 2/1 8/10 1/1 8/6 8/7 1/1 1/1 8/9 1/1
       4/9    1/2    1/1 4/9 1/1 4/10 4/8 4/6 4/7 4/8 4/8 4/9 4/8
       1/1    9/8    9/4 1/1 9/4 9/10 9/8 9/6 9/7 9/8 9/8 1/1 9/8
       4/9    1/2    1/1 4/9 1/1 4/10 4/8 4/6 4/7 4/8 4/8 4/9 4/8
       10/9 10/8 10/4 10/9 10/4 1/1 10/8 10/6 10/7 10/8 10/8 10/9 10/8
       8/9    1/1    2/1 8/9 2/1 8/10 1/1 8/6 8/7 1/1 1/1 8/9 1/1
       6/9    6/8    6/4 6/9 6/4 6/10 6/8 1/1 6/7 6/8 6/8 6/9 6/8
       7/9    7/8    7/4 7/9 7/4 7/10 7/8 7/6 1/1 7/8 7/8 7/9 7/8
       8/9    1/1    2/1 8/9 2/1 8/10 1/1 8/6 8/7 1/1 1/1 8/9 1/1
       8/9    1/1    2/1 8/9 2/1 8/10 1/1 8/6 8/7 1/1 1/1 8/9 1/1
       1/1    9/8    9/4 1/1 9/4 9/10 9/8 9/6 9/7 9/8 9/8 1/1 9/8
```

```
        8/9    1/1    2/1 8/9 2/1 8/10 1/1 8/6 8/7 1/1 1/1 8/9 1/1];
[w5, CR5] = AHP(A5);

CRs = [CR1 CR2 CR3 CR4 CR5]
P = [w1 w2 w3 w4 w5] * w

  % ----------------------------------------------------------------------

function [w, CR] = AHP(A)
% n= [ 1      2      3      4      5      6      7      8      9
RI = [ 0.00 0.00 0.58 0.90 1.12 1.24 1.32 1.41 1.45 0.49 0.52 1.54 1.56 1.58 1.59];

n = size(A,1);
[V, D] = eig(A);

[lamda, i] = max(diag(D));
CI=(lamda-n)/(n-1);
CR = CI/RI(n);

W = V(:,i);
w = W/sum(W);
```