Problem Chosen
D

2020
MCM/ICM
Summary Sheet

Team Control Number
2003684

PnP: A Real-Time Passing Network - Player Combining Model

Summary

Soccer team the Huskies requires a mathematical model to evaluate the players' performance on teamwork and help the coach make structural strategies.

To address the issue, the paper proposes the real-time passing network - player combining model (PnP) that measures both players' individual and collective performances, which can help the coach decide the starting team formation before the match and make instant adjustments during the match.

The PnP model consists of three sub-models: static passing network, player model and real-time passing network.

The static passing network is a weighted directed graph where nodes are players and edges represent distance. The weight of edges involves the type and length of passes as well as players' positions. Pass clustering coefficient , a modification of clustering coefficient in graph theory, is introduced to measure a player's participation and indicate team performance. K-means clustering algorithm is applied to divide players into three levels. Finally the pass clustering coefficient is compared with the PageRank value to identify the better model.

The player model is a comprehensive evaluating system, inspired by football games. It takes four major scores into account, namely shot, pass, defense and body. Each score contains several subscores computed by certain indices derived from raw data. Then, Analytic Hierarchy Process is employed to determine the weights of four major scores in different positions. These four scores form the total score. According to players' total scores in different positions, we define their types: offensive, defensive or organizational.

Based on the player model and dyadic or triadic configuration given by the model, three different sets of team formation, respectively focusing on attack, defense, and organization, are recommended to the coach.

The real-time passing network is a modification of the static passing network. Given a time parameter, it is able to screen out the statistics in certain time period and apply the same assessment as the static passing network to them.

Based on the player model and dyadic or triadic configuration given by the model, three different sets of team formation, respectively focusing on attack, defense and organization are recommended to the coach. Besides, combined with player model, the real-time passing network is able to give the coach suggestions about substitutions. By analyzing the real-time passing network of the opponent and applying the min-cut algorithm, the model can provide the coach with more real-time defense strategies towards the opponent. The PnP model can also be generalized to improve teamwork in societies through proper analogy.

The end of the paper represents the sensitivity analysis and discussion of strengths and weaknesses.

Keywords: clustering coefficient; PageRank; K-Means; AHP; min-cut

PnP: A Real-Time Passing Network - Player Combining Model

February 17, 2020

Contents

1 Introduction

1.1 Problem Background

In an interconnected society, it requires the coordination between people with diverse expertise to address complex issues. Competitive team sports like soccer are good illustrations of the situation. A team will function well only if players can effectively cooperate with teammates and fully take advantage of their features.

Coach is commonly considered as the person who decides how the ball should pass through players, but we should not neglect the role of data in the decision. Most European football clubs in premier leagues have data analysts in the coaching staff nowadays to evaluate players' performances and give advice to the coach. Mathematical model evaluation even becomes the determinant when Liverpool FC decides whether to sign a player [1]. Therefore, it is necessary to devise a model to understand the team's dynamics and figure out the best configuration of the team.

1.2 Our Work

A successful coach should be good at both the tactical arrangement before the match and the on-the-spot adjustments, therefore, we consider the problem from two aspects:

- To make proper arrangements, we need to evaluate the performance of the Huskies based on the historical data. The problem can be solved on two levels: teamwork and individual ability. In Section 4.1, we establish a passing network to evaluate the players' coordination mainly based on the passing data. Besides, teamwork should be the correct combinations of players with different features, so we devise a player model in Section 4.2 to evaluate each player's ability using the events data.
- For on-the-spot adjustment, we modify the passing network, proposing real-time passing network in Section 4.3, which can dynamically measure a player's performance during the match.

After the establishment of real-time passing network - player combining model, we show its effectiveness in Section 5, further discuss the model in Section 6 and make conclusions in Section 7.

2 Assumptions

1. **Pass routes and acceleration routes are all straight lines.**

   Admittedly, pass routes and acceleration route can be curves sometimes but generally speaking, straight pass and acceleration play a dominant role. Therefore, we can safely regard all of them as straight routes.

2. **Successful duels and pass mean those followed by teammate's action.**

   It is reasonable that a player's duels and pass are considered successful if his teammates control the ball after his action.

3. **A player shoots in target when the goal keeper makes save attempts.**

   There is no doubt that the goal keeper will make save attempts when a shot is within the goal range.

4. **Player scores hardly change in the next season.**

   Since we have analyzed players' performance consistently throughout the last season, dramatic changes in player scores will not happen in the next season.

3 Notations

The following table shows the notations we use in the model. We will clarify the definitions when the corresponding concepts first appear in the main body.

Symbol	Definition
G	Passing network
V	Set of nodes in the passing network
E	Set of edges in the passing network
Δt	Time period in real-time passing network
$G_r(\Delta t)$	Real-time passing network with parameter Δt
$V_r(\Delta t)$	Set of nodes in the real-time passing network with parameter Δt
$E_r(\Delta t)$	Set of edges in the real-time passing network with parameter Δt
n	Number of players in the passing network
$\overline{x_i}$	Average position of player i on x axis
$\overline{y_i}$	Average position of player i on y axis
$N_{i,j}$	Number of passes from player i to j
$d^t_{i,j}$	Pass times distance from player i to j
$d^l_{i,j}$	Average pass length from player i to j
$d^c_{i,j}$	Average coordinate distance between player i and j
$w_{i,j}$	Weight of the edge from player i to j in the network
c_i	Pass clustering coefficient of player i in the network
$\overline{c_C}$	Average clustering coefficient of players in Group C
S^t_i	Total score of player i
S^b_i	Body score of player i
S^d_i	Defense score of player i
S^p_i	Pass score of player i
S^s_i	Shot score of player i

4 Statement of our Model

In this section, we will present all the details of PnP model. Using historical data of the 38 games, we establish a passing network model and player model to evaluate the players' collective and individual performance. Then we expand the passing network, deriving a real-time passing network, which can measure the players' current performances and help the coach make on-the-spot adjustments.

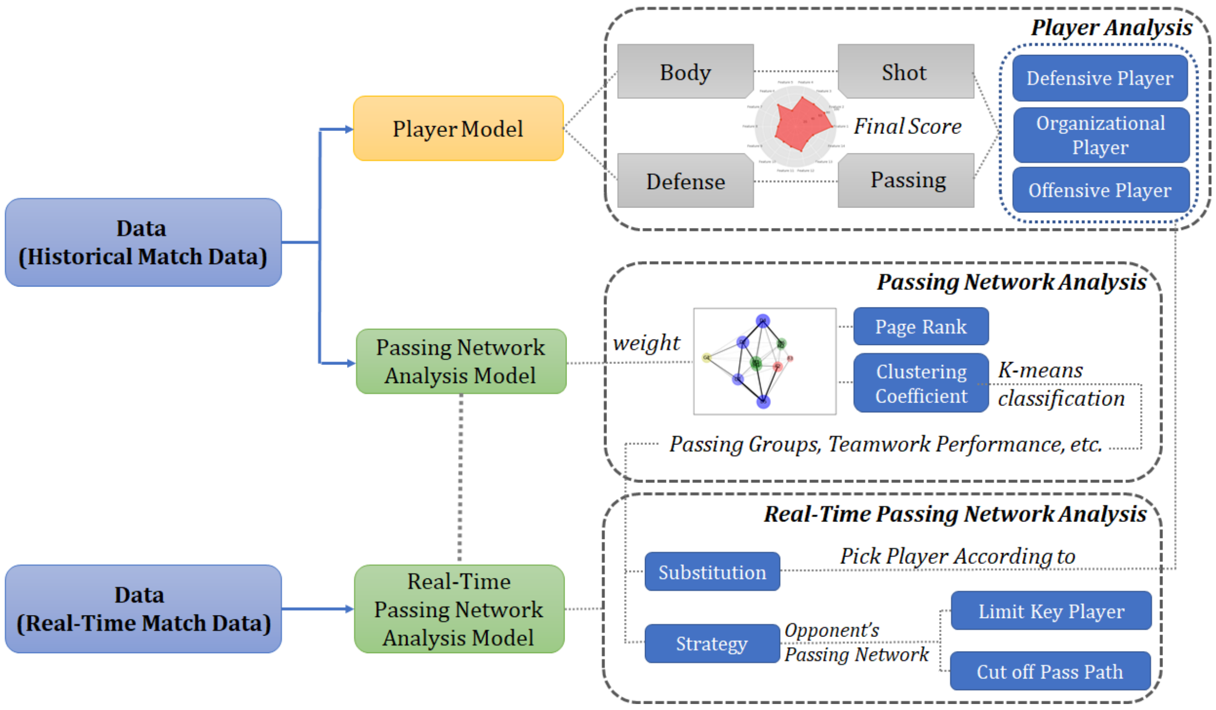


Figure 1: The overview of PnP model

4.1 The Passing Network

A key factor that reflects the coordination between players is the distribution of passes. We devise a basic passing network to show the team's overall passing performance in one match.

The passing network is a **weighted directed graph** $G = (V, E)$, where one node represents a player and an edge $(u \to v, w)$ is a pass route from player u to player v. The weight of the edge w will be calculated in the following subsection.

4.1.1 Weight of Edges in the Passing Network

The weight of the edges shows the distance between players, which is a combination of three different kinds of distances considering multiple features like the type of passes and players' positions.

- **Pass times distance**

  Based on the given data, we can obtain $N_{i,j}$, the number of passes from player i to j in one match. If the number of passes between two players is larger, then their positions are more likely to be closer, since passes between defenders are much more than that from a defender to a forward. Therefore, we define the **pass times distance** from player i to player j as:

$$d^t_{i,j} = \begin{cases} 0, & \text{if } i = j \\ \frac{1}{N_{i,j}}, & \text{if } i \neq j, N_{i,j} \neq 0 \\ +\infty, & \text{if } i \neq j, N_{i,j} = 0 \end{cases} \tag{1}$$

- **Average pass length**

  We can also get the length of passes from the data. To show the length of passes, we define the **average pass length vector** as:

$$\overline{D^l_{i,j}} = \left[\overline{d^{head\ pass}_{i,j}}, \overline{d^{hand\ pass}_{i,j}}, \overline{d^{high\ pass}_{i,j}}, \overline{d^{simple\ pass}_{i,j}}, \overline{d^{launch}_{i,j}}, \overline{d^{cross}_{i,j}}, \overline{d^{smart\ pass}_{i,j}}\right]$$

  where $\overline{d^{Type}_{i,j}}$ is the average pass length of each type from player i to j in one match. The length of each pass is the Euclidean distance between the origin and the destination of the pass.

  Simply combining the items of average pass length vector together cannot reflect the distance precisely, because the type of pass can reflect the distance between players to some extent. For example, a smart pass can give us more information about passing distance than a simple pass, and a high pass may provide more. As a result, we give the pass length coefficient vector:

$$\begin{aligned} K^l &= \left[\overline{k^{head\ pass}}, \overline{k^{hand\ pass}}, \overline{k^{high\ pass}}, \overline{k^{simple\ pass}}, \overline{k^{launch}}, \overline{k^{cross}}, \overline{k^{smart\ pass}}\right]^T \\ &= \frac{1}{6.7}[0.8, 0.6, 1.3, 1.0, 0.7, 1.2, 1.1]^T \end{aligned}$$

  The **average pass length** from player i to player j is:

$$d^l_{i,j} = \begin{cases} \overline{D^l_{i,j}} K^l, & \text{if } N_{i,j} > 0 \\ +\infty & \text{if } N_{i,j} = 0 \end{cases} \tag{2}$$

- **Average coordinate distance**

  The previous two kinds of distances only consider passes, which is not informative enough as the number of passes between two players may be limited. We take the average coordinates of player i $(\overline{x_i}, \overline{y_i})$ into account. The **average coordinate distance** between player i and j is the Euclidean distance between their average positions:

$$d^c_{i,j} = \sqrt{(\overline{x_i} - \overline{x_j})^2 + (\overline{y_i} - \overline{y_j})^2} \tag{3}$$

With Equation (1), Equation (2), and Equation (3), the **weight of the edge** from i to j is calculated by:

$$w_{i,j} = \exp\left(\frac{1}{k_c \overset{\sim}{d^c_{i,j}} + k_l \overset{\sim}{d^l_{i,j}} + k_t d^t_{i,j}}\right) \tag{4}$$

where:

- $k_t = 0.7, k_l = 0.1, k_c = 0.2$
- $\overset{\sim}{d^l_{i,j}} = \frac{d^l_{i,j}}{\max\limits_{i,j}\{d^l_{i,j}\}}, \overset{\sim}{d^c_{i,j}} = \frac{d^c_{i,j}}{\max\limits_{i,j}\{d^c_{i,j}\}}$

Figure 2 shows the passing network of match 1. The size of nodes is related to the number of passes launched by the player while the thickness of edges is related to the weight.

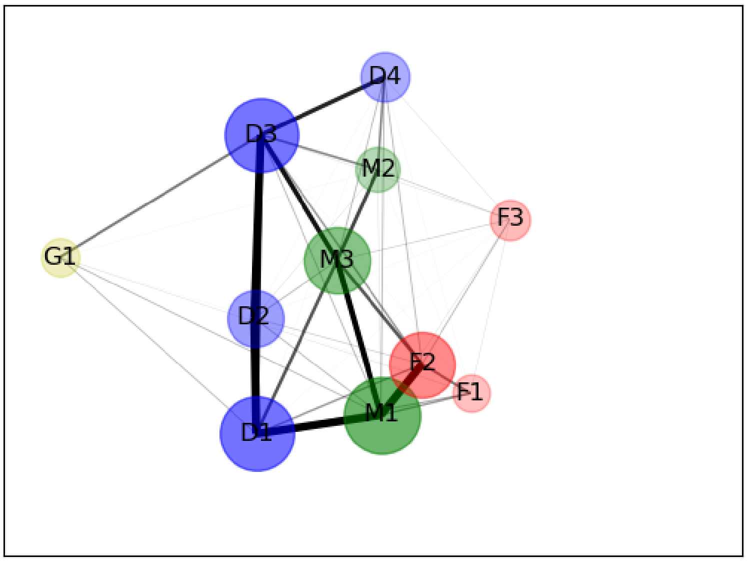


Figure 2: The passing network of match 1

4.1.2 Pass Clustering Coefficient in the Passing Network

Now we introduce **pass clustering coefficient**, which is a modification of the clustering coefficient in graph theory. Its original definition can be found in paper [2], measuring how well connected a node is to other nodes in the graph. Similarly, a high pass clustering coefficient in our network indicates a well-connected player in the team.

A non-weighted adjacency matrix $a_{i,j}$ is given here for convenience:

$$a_{i,j} = \begin{cases} 0, & \text{if } i = j \quad \text{or} \quad i \neq j, N_{i,j} = 0 \\ 1, & \text{if } i \neq j, N_{i,j} > 0 \end{cases}$$

where $N_{i,j}$ is the number of passes from player i to j.

Then the **degree** k_i of a player(node) is $\sum_{i=1}^{n} a_{i,j}$.

Finally, we define the **pass clustering coefficient** of player i as:

$$c_i = \begin{cases} \frac{1}{k_i(k_i-1)} \sum_{j,h} \frac{w_{j,i}+w_{i,h}}{2} a_{j,i} a_{i,h} a_{j,h}, & \text{if } k_i \geq 2 \\ 0, & \text{if } k_i < 2 \end{cases} \tag{5}$$

To evaluate the effect of player i in the team, we consider a situation in which player j wants to pass the ball to h but cannot pass directly. Such situations often happen because of the opponents' defense or the far distance. As a result, player i acts as an intermediary, forming a pass route $j \rightarrow i \rightarrow h$. The average weight of the route $\frac{w_{j,i}+w_{i,h}}{2}$ in Equation (5) reflects the engagement of player i in the process. However, such route may not exist, so we introduce the adjacency indicator $a_{j,i}a_{i,h}$ to ensure its existence . Besides, if $a_{j,h} = 0$, then it shows that player j can not directly pass the ball to h, he has no choice but to pass the ball to another player. This kind of situation does not reflect active teamwork and is eliminated by the introduction of $a_{j,h}$. The term $\frac{1}{k_i(k_i-1)}$ is based on the original definition of clustering coefficient in paper [2].

4.1.3 Pass Clustering Coefficient versus PageRank

The pass clustering coefficient c_i can effectively measure one player's contribution to the team, making distinct difference between various levels of players. In fact, our initial thought about the passing network is a much simpler model, where the weight of the edges is just the number of passes and players are evaluated by the classic **PageRank** algorithm.

PageRank is an algorithm to evaluate the influence of websites based on the hyperlinks between them. First proposed by Larry Page in 1999 [3], PageRank algorithm becomes the core of the search engine giant, Google. The basic idea of PageRank is that if there are many hyperlinks point to a page, then the page should be influential and have a high PR(PageRank) value. Meanwhile, those pages that a page with high PR value points to should also also be influential and have high PR value. This idea generates from the impact factor of papers, as the most cited papers are most influential and have higher impact factors. The algorithm can also measure the influence of a node in the network, like a player's role in the team. We will apply both our passing network and the PageRank algorithm to the data of one match to show that the passing network suits the problem better.

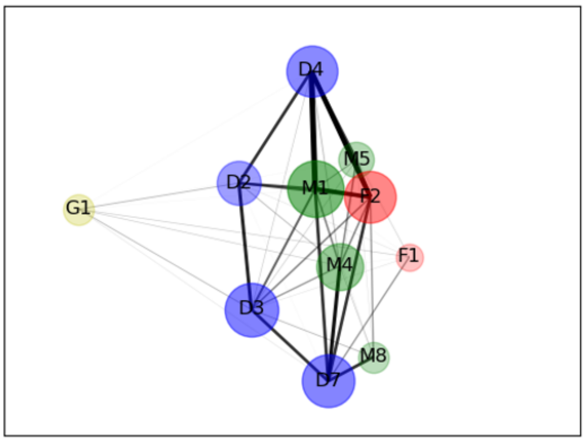
A mathematical method to find out the PR value is to calculate the eigenvector of the adjacency matrix $N_{i,j}$, but a more common-used method proposed in Larry's paper [3] is the **RandomWalk** model. A surfer will randomly jump from one page to another through hyperlinks. With the increase of visits, the frequency he visits a certain page will converge to a fixed value, which is defined as the PR value of the page. By iterating the frequency vector for a sufficient large of times, we can get the PR value of each page. We calculate the PR value of players based on the same idea. The PR value of player x is given by:

$$PR_x = (1-d) + d \sum_{(y \to x, w)} \frac{w}{\sum_{(y \to z, w')} w'} PR_y$$

where $d = 0.95$ and we iterate the vector for 50 times. The code of the PageRank algorithm can be found in the Appendix B.

Algorithm 1: PageRank($epoch, d$)

```
foreach $x \in V$ do
    $pr_x \leftarrow 0$ $sum_x \leftarrow 0$
end
foreach $x, y \in V, (x \to y, w) \in E$ do
    $sum_x \leftarrow sum_x + w$
end
while $epoch > 0$ do
    $last \leftarrow pr$
    foreach $x \in V$ do
        $res \leftarrow 0$
        foreach $y \in V, (y \to x, w) \in E$ do
            $res \leftarrow res + w/sum_y * pr_y$
        end
        $pr_x \leftarrow (1-d) + d * res$
    end
    $epoch \leftarrow epoch - 1$
end
return pr
```



Player	PageRank
F2	**2.025457**
D7	1.840451
D4	**1.488551**
M1	**1.452186**
M4	1.175
D3	0.915025
M5	0.844812
D2	0.805151
F1	0.765173
M8	0.713502
G1	0.423367

Player	Clustering Coefficient
F2	**441.742**
M1	**378.5591**
M8	226.569
D7	209.8493
D4	**152.2017**
D2	38.6647
M5	29.56074
D3	28.00758
M4	26.69309
G1	11.16222
F1	3.951268

Figure 3: The pass clustering coefficient and PR value of players in match 10

Figure 3 shows the clustering coefficient c_i and PR value of players in match10. We can see the distribution of c_i covers a much larger range than the PR value, which will be very helpful for classifying players into three levels later. It can also be observed that defenders rank lower when they are measured by c_i, because we take into account the types of passes and the positions of players. Many passes of defenders are of minor importance as the ball is just hanging around in the backfield, not contributing to the team's coordination and goals. Compared with PageRank, the clustering coefficient is more in line with reality; it works better to evaluate a player's performance on teamwork.

4.1.4 Classifying Players by K-Means Clustering Analysis

With the pass data of a whole game, we can calculate each player's pass clustering coefficient c_i. However, it is a local indicator that only measures one player. To obtain the overall information of the team, we study the average c_i of all players and find that it is **positively related to** the result of the game. Meanwhile, we should consider the players' individual performance in the game to help coach make decisions.

To address these problems, we first divide the players into 3 groups A, B and C according to their c_i in one match using **K-Means Cluster Analysis**. Note that these groups show relative relationships between players. For example, the average c_i of Group A may vary greatly among different matches. Then we focus on the average c_i of group C $\overline{c_C}$ and the number of players in Group A and Group C, which turn out to determine the result to a great extent.

K-Means clustering analysis is an unsupervised learning algorithm. The number of clusters K is set manually. First, the algorithm randomly chooses K samples from the dataset as the centers of the clusters. Then for each sample x_i, it will be distributed to the cluster whose center is closest to x_i. After this step, we will recalculate the center of each cluster. By repeating the two steps above until the centers of clusters do not change, the algorithm is finished.

We apply this algorithm to the players' pass clustering coefficients to evaluate their performances on teamwork in one match. Since different players' c_i may vary by 10 times (as shown in Figure 3), directly dividing the players into 3 clusters is not practical. Therefore, we first use K-Means to classify the players into 2 groups, Group A and others. Group A involves the players with the highest pass clustering coefficients, indicating that they are

Algorithm 2: K-Means($K, list, \varepsilon$)

```
Randomly choose $K$ distinct items as initial $Center$: $Center_0, Center_1, ..., Center_{K-1}$
while true do
    foreach $x \in list$ do
        $dis \leftarrow +\infty$
        $pos = -1$
        for $i = 0$ to $K$ do
            if $Distance(x, Center_i) < dis$ then
                $dis \leftarrow Distance(x, Center_i)$
                $pos \leftarrow i$
            end
        end
        Insert $x$ in $BelongList_{pos}$
    end
    for $i = 0$ to $K$ do
        $NewCenter_i \leftarrow$ Average $(BelongList_i)$
    end
    if $\forall i$, $Distance(NewCenter_i, Center_i) < \varepsilon$ then
        break
    end
end
return $Center$
```

the key players in the network. We execute the same operation on the rest players, dividing them into Group B and C, where Group C consists of players with the lowest c_i. The average pass clustering coefficient of each group can give us information on the whole team's performance. We particularly focus on the **average pass clustering coefficient of Group C** $\overline{c_C}$, as we think that the players with the worst performance will be the chance for opponents' attack. In fact, we find $\overline{c_C}$ is **positively related to** the result of the game, which will be further discussed in Section 5.

4.2 Player Model

Besides the players' performance on teamwork, we also have to capture every player's individual abilities to arrange different formations according to the real situation. Therefore, we use the data provided to analyze both the player score and the player type.

4.2.1 Player Score

With the help of the data, we divide a player's ability into the following four aspects: **body**, **defense**, **pass**, **shot**. Figure 4 demonstrates the specific components of the player score. We have to mention that penalty is excluded from the shot score because we think it seldom happens and the kicker remains the same. Consequently, the penalty data has little influence on players' shot scores. What's more, we take no account of the goal keeper due to the difference of evaluating criteria between goal keepers and other players. For lack of the goal keeper's information, we can hardly figure out the real ability of the goal keeper. Moreover, goal keeper G1 is in the starting line-up of every match, indicating his priorities over other goal keepers.

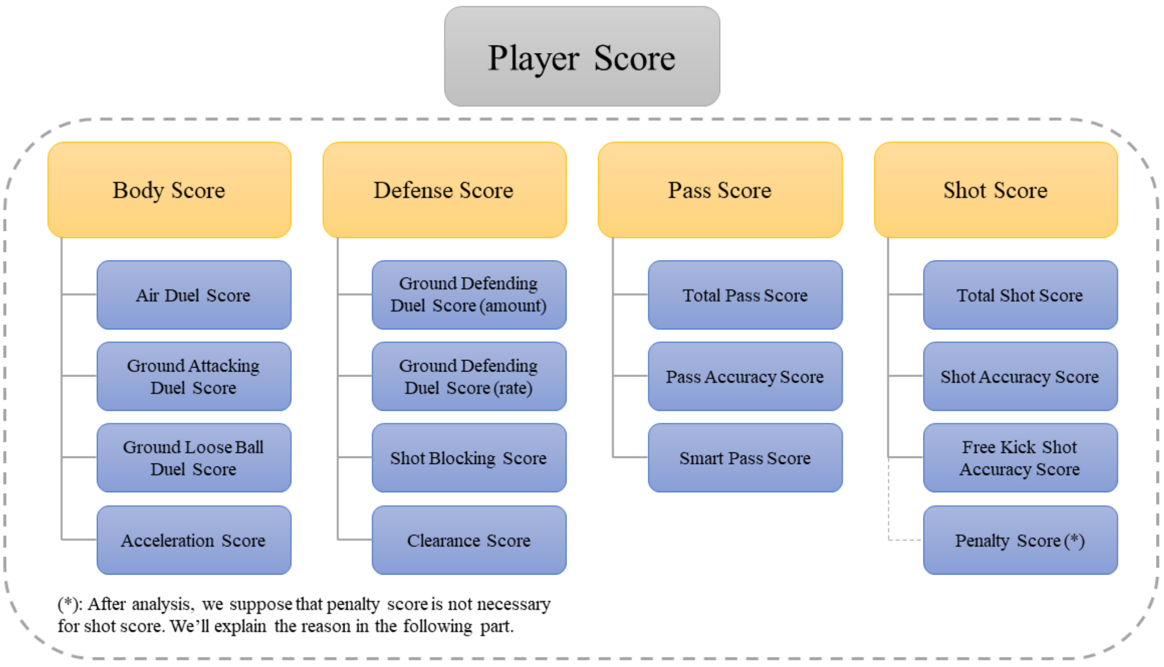

Player Score
Body Score
Air Duel Score
Ground Attacking Duel Score
Ground Loose Ball Duel Score
Acceleration Score
Defense Score
Ground Defending Duel Score (amount)
Ground Defending Duel Score (rate)
Shot Blocking Score
Clearance Score
Pass Score
Total Pass Score
Pass Accuracy Score
Smart Pass Score
Shot Score
Total Shot Score
Shot Accuracy Score
Free Kick Shot Accuracy Score
Penalty Score (*)
(*): After analysis, we suppose that penalty score is not necessary for shot score. We'll explain the reason in the following part.

Figure 4: The components of the player score

For every sub-score of the player i, we find a decisive index id and apply the same normalization formula to get a sub-score $\hat{s}_i^{id}$ between 0 to 1. Then, we just map $\hat{s}_i^{id}$ into s_i^{id} , which ranges from 40 to 100, with Equation (6).

$$\hat{s}_i^{id} = \frac{id_i - \min\limits_i id_i}{\max\limits_i id_i - \min\limits_i id_i}$$

$$s_i^{id} = k\hat{s}_i^{id} + b \tag{6}$$

where $k = 60, b = 40$.

Getting the sub-score, we determine the weight of each sub-score and integrate them into four main scores through the formula:

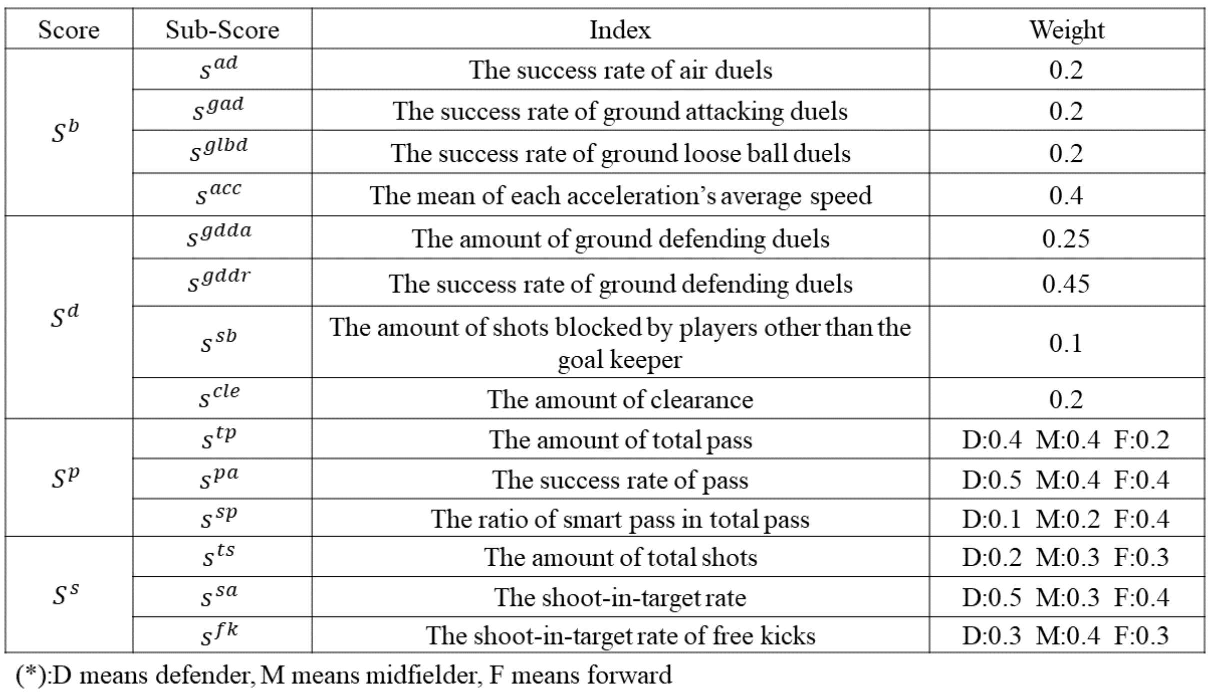
$$S_i^{ability} = \sum_{id} w_{id} s_i^{id}$$

For simplicity, we list the index and weight of every sub-score in Figure 5. After obtaining the four main scores, we use Equation (7) to calculate the total score.

$$S_i^t = w_b S_i^b + w_d S_i^d + w_p S_i^p + w_s S_i^s$$

To set the weight in Equation (7), we apply **Analytic Hierarchy Process(AHP)** [4]. AHP is an especially effective way to decide on weight distribution when expert knowledge is provided. First, we construct the judgement matrix for defenders, midfielders and forwards.

$$J_D = \begin{bmatrix} 1 & 3 & \frac{3}{2} & \frac{2}{3} \\ \frac{1}{3} & 1 & \frac{1}{2} & \frac{1}{4} \\ \frac{3}{4} & 2 & 1 & \frac{1}{2} \\ \frac{3}{2} & 4 & 2 & 1 \end{bmatrix} J_M = \begin{bmatrix} 1 & 2 & 2 & \frac{3}{2} \\ \frac{2}{3} & 1 & \frac{3}{2} & 1 \\ \frac{1}{2} & \frac{2}{3} & 1 & \frac{3}{4} \\ \frac{2}{3} & 1 & \frac{3}{2} & 1 \end{bmatrix} J_F = \begin{bmatrix} 1 & \frac{1}{2} & 2 & 3 \\ 2 & 1 & 3 & 4 \\ \frac{2}{3} & \frac{1}{3} & 3 & \frac{3}{2} \\ \frac{1}{2} & \frac{1}{4} & \frac{2}{3} & 1 \end{bmatrix}$$



Score	Sub-Score	Index	Weight
S^b	s^{ad}	The success rate of air duels	0.2
	s^{gad}	The success rate of ground attacking duels	0.2
	s^{glbd}	The success rate of ground loose ball duels	0.2
	s^{acc}	The mean of each acceleration's average speed	0.4
S^d	s^{gdda}	The amount of ground defending duels	0.25
	s^{gddr}	The success rate of ground defending duels	0.45
	s^{sb}	The amount of shots blocked by players other than the goal keeper	0.1
	s^{cle}	The amount of clearance	0.2
S^p	s^{tp}	The amount of total pass	D:0.4 M:0.4 F:0.2
	s^{pa}	The success rate of pass	D:0.5 M:0.4 F:0.4
	s^{sp}	The ratio of smart pass in total pass	D:0.1 M:0.2 F:0.4
S^s	s^{ts}	The amount of total shots	D:0.2 M:0.3 F:0.3
	s^{sa}	The shoot-in-target rate	D:0.5 M:0.3 F:0.4
	s^{fk}	The shoot-in-target rate of free kicks	D:0.3 M:0.4 F:0.3

(*):D means defender, M means midfielder, F means forward

Figure 5: The index and weight of each sub-score

where the row(column) order is pass, shot, body and defense respectively.

Then, we normalize the three matrices and calculate the eigenvalues and feature vectors of them. Subsequently, we use the following two equations to calculate **Consistency index(CI)** and **CR**.

$$CI = \frac{\lambda_{max} - n}{n - 1}$$

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

where $n = 4$, $RI = 0.9$, and λ_{max} means the maximum of all the eigenvalues.

Therefore, we get $CR_D = 0.0118$, $CR_M = 0.0425$, $CR_F = 0.0738$, which are all less than 0.1 , thus passing the consistency test. Finally, we get all the weights listed below in Table 1

Table 1: AHP result

Position	Pass Weight	Shot Weight	Body Weight	Defense Weight
Defender	0.288724	0.099053	0.204074	0.408148
Midfielder	0.368227	0.233279	0.165215	0.233279
Forward	0.274037	0.45768	0.156978	0.111305

We plug these weights into Equation (7) . With Equation (7), we calculate and depict all the players' scores in different positions in Figure 6. If a player's any main score exceeds 80, it will be highlighted in green, which means this main score of the player has an obvious superiority over others'. Besides, the bold digit suggests that a player may be more competent in one position than others.

$$\begin{cases} S_{i,D}^t = 0.29S_i^p + 0.1S_i^s + 0.2S_i^b + 0.4S_i^d, \\ S_{i,M}^t = 0.37S_i^p + 0.23S_i^s + 0.17S_i^b + 0.23S_i^d, \\ S_{i,F}^t = 0.27S_i^p + 0.46S_i^s + 0.16S_i^b + 0.11S_i^d, \end{cases} \tag{7}$$

Player	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	F1	F2	F3	F4	F5	F6	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13
Body	75	69	72	76	76	73	73	79	76	61	73	72	57	75	65	75	74	70	77	78	**84**	72	51	78	83	67	61	72	61
Defense	**80**	79	72	50	71	59	57	51	53	66	58	64	44	69	64	51	75	56	66	57	56	56	57	58	42	44	47	60	56
Pass	**81**	78	79	72	75	60	62	60	50	45	58	77	46	68	68	57	**89**	47	**80**	70	46	70	49	52	56	45	47	47	60
Shot	52	55	55	67	52	53	60	50	49	47	71	75	53	76	64	71	60	57	47	55	53	73	49	55	53	47	47	58	48
Forward	67	66	66	**68**	64	59	62	57	54	51	**66**	**74**	**51**	**73**	**65**	**66**	73	56	63	63	56	**70**	50	58	57	49	49	58	54
Defender	**77**	**74**	**72**	63	**71**	**61**	**62**	**59**	**56**	**57**	62	70	48	70	65	59	**78**	**56**	**70**	64	**58**	65	52	**60**	55	49	**50**	**58**	**57**
Midfielder	**74**	71	70	**66**	69	60	62	59	55	53	63	**73**	49	71	65	62	**77**	55	**68**	**64**	**56**	**68**	**51**	**58**	**56**	**49**	**49**	**57**	**56**

Figure 6: The index and weight of each sub-score

In short, Figure 6 gives us a comprehensive view on players' abilities and compares those scores in both personal and interpersonal perspective.

4.2.2 Player Type

After obtaining all the players' scores in different positions, we can further classify them into three types: **offensive, organizational** and **defensive** players. This is consistent with the bold digits in Figure 6. To provide an intuitive view, we depict three player's scores in Figure 7. Those three players perform best compared with others in the same position. As we can see, they tend to have a relatively high score in any of the four domains. According to their features, we categorize D1 into defensive and organizational defender, M1 into organizational and defensive midfielder and F2 into offensive and organizational forward. For other players, we define their type in the same method. You can refer to Appendix D for details.

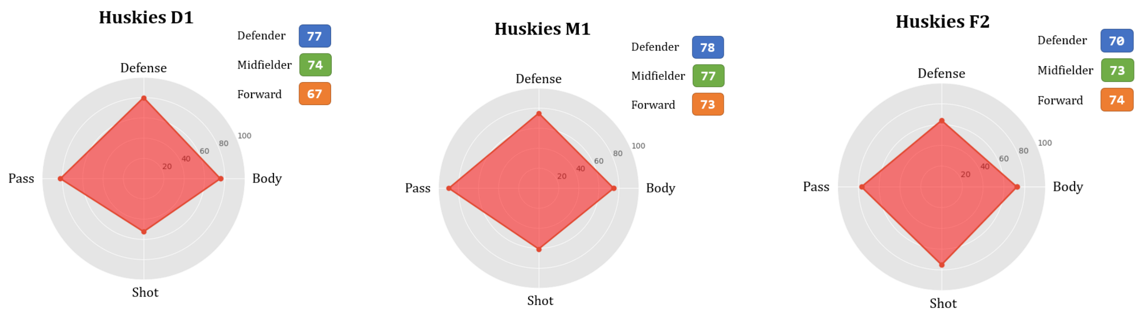


Figure 7: Scores of three players

With all the players' scores and types, we can offer the coach a deeper knowledge on the players and help him to form strategies either before or during a match, which will be discussed in Section 5.2.

4.3 RTPN: Real-Time Passing Network

4.3.1 Definitions of RTPN and Real-Time Pass Clustering Coefficient

After building the static passing network of a match, we are able to establish a dynamic one, which is named "**Real-Time Passing Network**" or **RTPN** in short. The RTPN is constructed by the data of last few moments. As a result, it indicates the current performance of every player. The network can also reflect the main passing group and formation of the team. With the help of the RTPN, we are able to analyze the data more dynamically.

To state formally, we define RTPN as a weighted directed graph $G_r(\Delta t)$, where Δt indicates the time period under consideration is $(t - \Delta t, t)$ supposing t stands for the current time. Like other graphs, $G_r(\Delta t)$ is constructed by its vertex set $V_r(\Delta t)$ and edge set $E_r(\Delta t)$. Its vertex set includes all the players who has participated in an event in the time period $(t - \Delta t, t)$. Similarly, its edge set includes all the pass route in this time period. The weight on the edge is computed with the same methods we stated above.

With the RTPN built, we are able to calculate the **pass clustering coefficient** in the network. We use the same definition in Section 4.1.2 and use the same methods to compute it. The real-time clustering coefficient data in match 1 is shown in Figure 8.

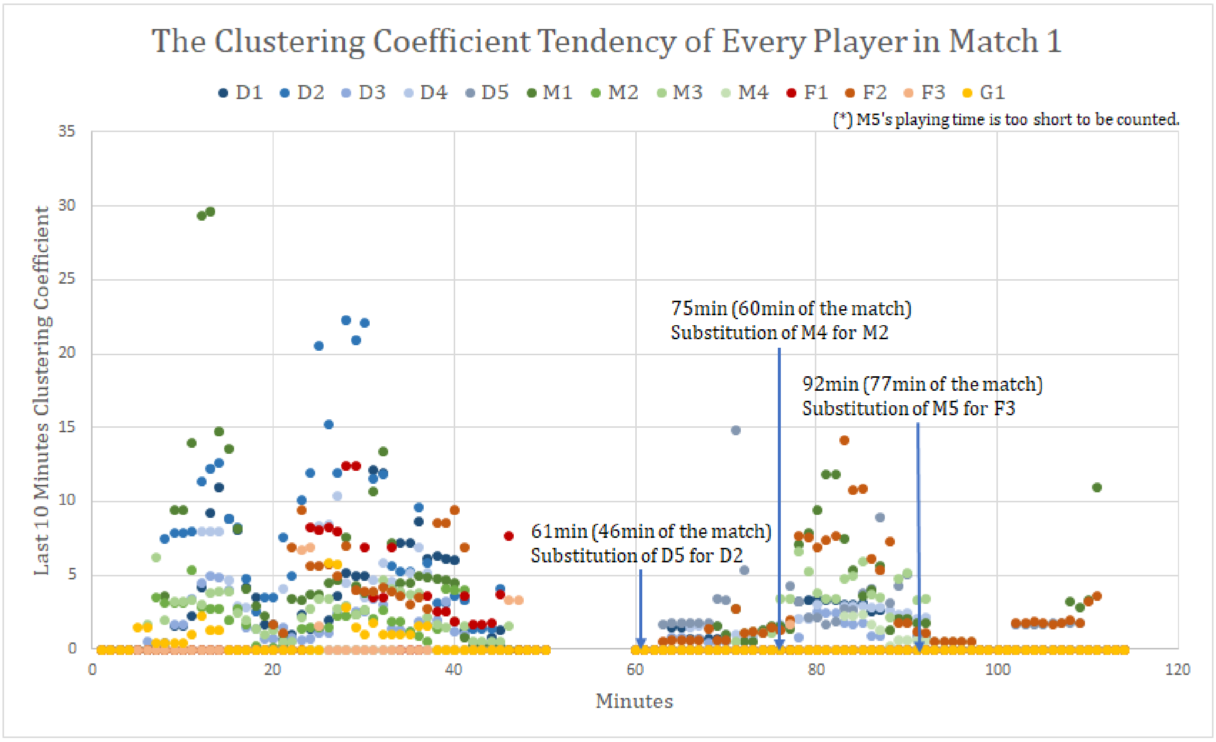


Figure 8: The real-time clustering coefficient tendency in match 1

We can infer a lot of information from this figure, which we will explain later in Section 5.3. In a word, like full-time clustering coefficient data, we can estimate the team performance and individual performance from it.

4.3.2 Real-Time Strategies based on RTPN

After we estimate the performance data, we are able to give team coach suggestions about substitutions. Combined with the player model we have mentioned above, we can help the coach to choose the substitutes more precisely.

Our RTPN model be used to analyze both our players' performances and opponent's, We proposed following **RTPN-based strategies**:

- **Attacking and Pass**: After estimating the performance data such as pass clustering coefficient in our RTPN, we are able to give team coach suggestions about substitutions. Combined with the player model we have mentioned above, we can help the coach to choose the substitutes more precisely.
- **Defense**: Use pass clustering coefficient in opponent's RTPN to analyze the core pass group of the opponent. If we could restrict their key player in the group, we might be able to reduce opponent's pass efficiency and control the game.

- **Defense**: The edge weights in our RTPN reflect the importance and frequency of the pass route. Sometimes it is not necessary and efficient to block all the heavy-weighted pass route. Since our goal is to prevent opponent from scoring, we believe if we can cut off the path from current ball-owner to the main forward, we are able to reduce opponent's goal to the utmost extent. The detailed methods will be explained in the next paragraph.

Here we are going to discuss the last strategy we provide above. Our goal is to cut the path from current ball-owner to main forward of the opponent. We regard current ball-owner as the source of RTPN and main forward as sink of RTPN. The weights of edge in the RTPN can also be denoted as the flow of the edge. Thus, our strategy is to find the minimum-cut of the flow network. That is, find a node set S satisfying:

$$\begin{cases} S \subseteq V_r(\Delta t) \\ source \in S \\ sink \in V_r(\Delta t) - S \end{cases}$$

while minimizing:

$$\sum_{(u \to v, w) \in E_r(\Delta t)}^{u \in S, v \in V_r(\Delta t) - S} w$$

The problem can be solved easily with enumerate algorithm in a time complexity of $O(|E_r(\Delta t)| \cdot 3^{|V_r(\Delta t)|})$, since $|V_r(\Delta t)|$ is small enough. If the size of $V_r(\Delta t)$ become larger, we can use the Max-flow min-cut theorem [5] to transform the problem into a maximum flow problem and use Dinic's Algorithm [6] to solve it in a time complexity of $O(|V_r(\Delta t)|^2 \cdot |E_r(\Delta t)|)$.

5 Implementation

With the accomplishment of the model construction, now we use the model to process the given data, demonstrating its effect. We will also propose corresponding strategies based on the analysis.

5.1 Average Clustering Coefficient

We calculate the **average clustering coefficient of Group C** $\overline{c_C}$ (introduced in Section 4.1.4) of all the 38 games through out the season, as shown in Figure 9.

We can observe a strong relationship between $\overline{c_C}$ and the result of the game. The team only lost one game among the games whose $\overline{c_C}$ is over 35 and only won one game among the 16 games whose $\overline{c_C}$ is under 15. It shows that more passes and less distinction between the players' performance are helpful to the team's victory.

However, for the games between this two levels, this indicator does not give a clear classification between victory and loss. The reason why this situation happens is that the passing network only processes the pass data. If a player is excellent at finishing but not

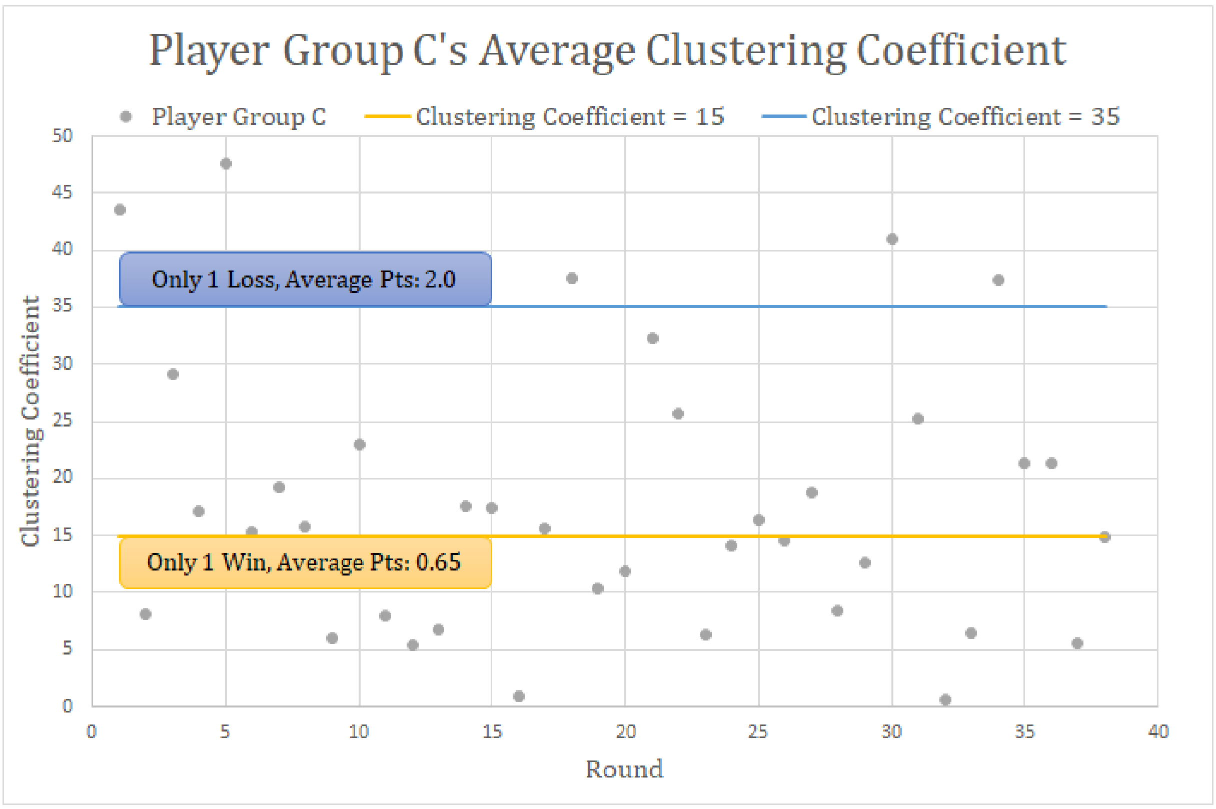


Figure 9: The relationship between $\overline{c_C}$ and the result

so good at organizing, like Inzaghi, his contribution to the team may be underestimated in the passing network because of the relatively lower number of passes. This problem can be greatly alleviated by introducing our player model as it considers the comprehensive ability of a player, so **the combination of passing network and player model** will be the best solution.

5.2 Pre-Match Structural Strategies

As is discussed in our model part, we can optimize our formation and player configuration either before or during a match with the analysis of individual model and passing network.

For lack of rival's concrete data, we cannot design a much too sophisticated strategy before a match; however, we can use our individual model to devise several sets of starting line-ups for different purposes. After researching every match's starting line-ups, we spot that there are altogether six sorts, which are 4-4-2, 4-3-3, 5-3-2, 4-5-1, 5-4-1 and 4-2-4, ranking in frequency from high to low. Combining individual abilities with players' preferred foot, we propose three distinct starting line-ups focusing on **organization (4-4-2)**, **attack (4-3-3)** and **defense (4-5-1)** respectively.

To determine the proper players, we should consider not only their individual ability but also their combination, so we study the players of Group A (the highest c_i)throughout the last season. The result shows that player M1 is the core of the team who launches most passes, while F2, M3 and M4 are also key players that dominate the passes. Moreover, there is a remarkable mutual pass relationship between either two or three of them. This **dyadic** or **triadic configuration** indicates their excellent teamwork.

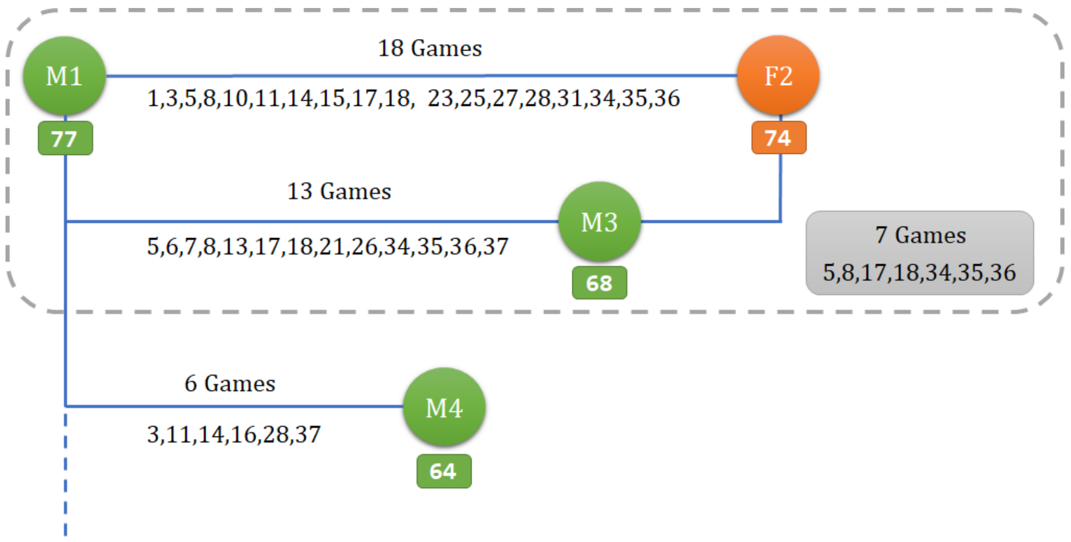


Figure 10: The players who make the best combination

As is depicted in Figure 11, we select eleven players for the organization formation based on all the factors mentioned above. We believe that those eleven players have stable performance and make good teamwork. This formation aims to seek for **a balanced trade-off** between attack and defense; meanwhile, it also elevates the **pass accuracy** and **number of times**.

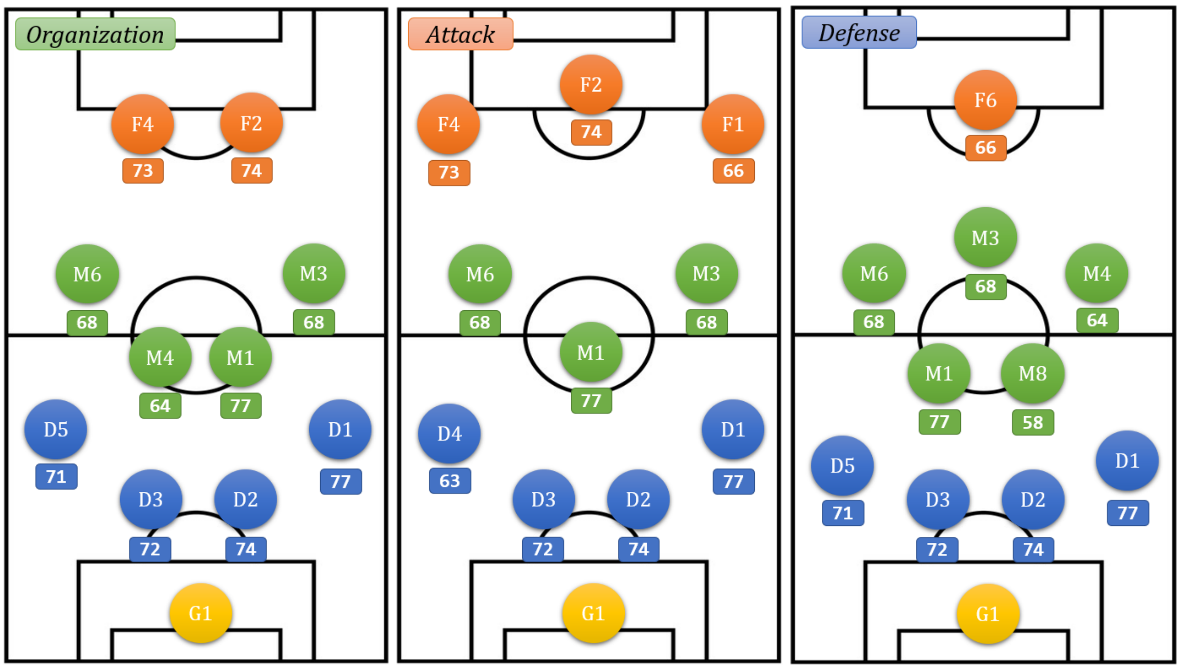


Figure 11: Three line-ups

For the attack formation, we replace M4 with F1 to improve the attacking capacity. Interestingly, we also substitute D4 for D5 though D5 possesses a higher score as a defender. We reckon that side defenders should also take an active part in attacking. Therefore, we choose D4, who belongs to **offensive defenders**, to reinforce our attacking ability.

For the defense formation, we add M8 to consolidate the defense of midfield. As you can see, we replace F2 and F4 with F6 because the acceleration score of F6 is the highest in forwards. When using the defense formation, we usually have to depend on quick counter-attacks to goal; hence, a forward with **outstanding acceleration capacity** may play a crucial role in the defense formation.

We recommend that the coach apply attack formation as the starting line-up to weaker rivals, organization formation to equivalent ones and defense formation to stronger ones.

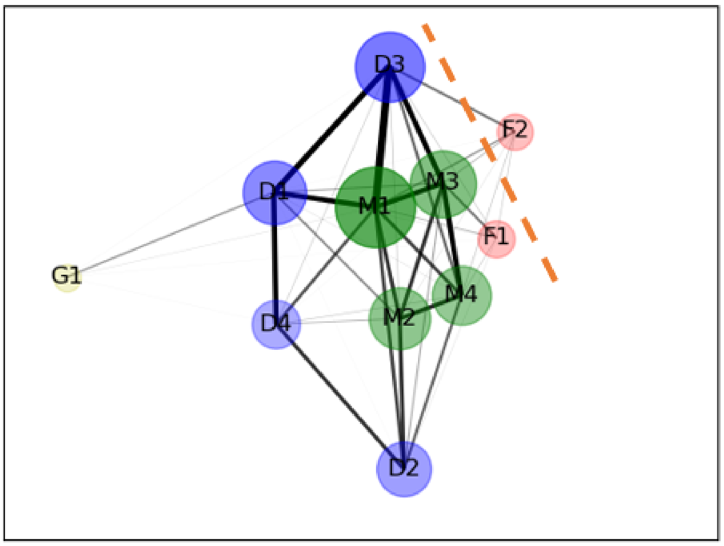
5.3 The Application of Real-Time Passing Network

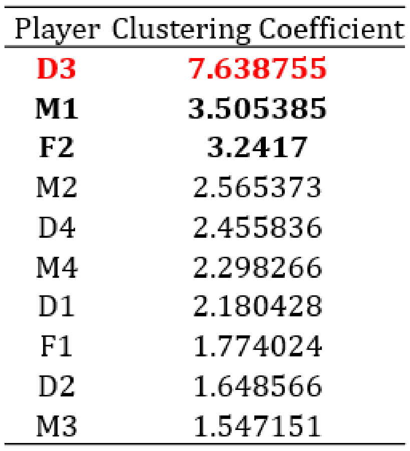
We have explained RTPN in Section 4.3.1. Here is an example of using RTPN to analyze match data. From Figure 8 we can infer the following conclusions:

- In the beginning of the match, player M1 and D1 had high clustering coefficient, which indicates they had formed a core dyadic pass relation. Based on their positions, we infer that during this period, what our team mainly did was defense.
- During the center period of the first half, player F1, F2, D4 began to join the passing group, which made a triadic pass relation of player M1, D1, F2. We can verify our conclusion from Figure 2. During this period, our team was mainly organizing and attacking.
- In the second half, the clustering coefficient of every player dropped owing to the lack of physical strength. Player M2, F1 and D4 began to have a high clustering coefficient, which indicates their attacking attempts were mainly in the wing side based on the average position in Figure 2. What we also know is the substitution of the coach came into effects immediately.

From the conclusions above, we suggests that our RTPN is able to assist the team coach in decision-making by assessing the player performance and main pass group.

Analyzing our opponent's data can help us have a better understanding of our opponent's structures and strategies, which is vital for us to set counter-strategies.





Player	Clustering Coefficient
D3	**7.638755**
M1	**3.505385**
F2	**3.2417**
M2	2.565373
D4	2.455836
M4	2.298266
D1	2.180428
F1	1.774024
D2	1.648566
M3	1.547151

Figure 12: The real-time passing network and pass clustering coefficient of opponent in match 5 (from 10 minutes to 20 minutes, the orange line stands for the min-cut plan)

For example, we use our RTPN to analyze the opponent's data in match 5 from 10 minutes to 20 minutes. The data is shown in Figure 12. From the statistics we can know that the current pass group of opponent is D3, M1 and F2, and D3 is the core of this group. In order to limit opponent's performance, we can take the following measures:

- Make some players cut off player D3's pass route, so that the group will disappear because player D3 is restricted.
- Keep the ball away from forward F2. Suppose the ball is controlled by D3 currently, then we can regard D3 as source and F2 as sink. Using our strategy in Section 4.3.2, we have to compute a min-cut from M1 to F2. After calculation, we find out the plan shown in Figure 12 can minimize the cost. So we can make some players cut off these paths, so that F2 can not get the ball easily. As a result, the opponent's performance will be limited.

In short, our model can be applied during the match, it can provides the team coach with more information about opponent and give useful suggestions.

5.4 Solutions to the tasks

- To identify the network patterns, we establish a passing network in Section 4.1 evaluating the team in one match. We also propose the real-time passing network in Section 4.3 to measure players' instant performance.
- The pass clustering coefficient c_i measures one player's coordination with teammates while the average clustering coefficient of Group C $\overline{c_C}$ can reflect the overall performance. More details can be found in Section 4.1.2 and Section 5.1.
- By applying the player model and static passing network to the given data, we propose the strategies on team formation in Section 5.2. According to the analysis of the passing network, we suggest that the midfielder and right wing should be reinforced in the next season. Also, the coach should emphasize on teaming skills to make the partial cooperation more diverse. Besides, the real-time passing network can help the coach with substitutions and real-time strategies, which is discussed in Section 4.3.2.
- Based on our PnP model, we are able to propose a universal model of team performance and individual performance. The model is a combination of three sub-models, **the cooperation network model**, **the individual model** and **the real-time cooperation network model**.
  - The first part, **the cooperation network**, which can indicate the total cooperation performance between every pair of team members, has a similar structure with our passing network. After redefining the weight of edges, we can also use PageRank and clustering coefficient to analyze individual performance, and use K-Means to divide team members into groups to analyze team performance and find out the core member of the group.
  - The second part, **the individual model**, which is based on all-time individual performance, is similar to our player model. We can use different indices, such as leadership, to analyze each member. After combining the indices together by certain methods like Analytic Hierarchy Process (AHP), we are able to find out the best position for every member in the team.

– The third part, **the real-time cooperation network model**, which is used to analyze the current team work performance of each member, is similar to our RTPN model. We reuse the definitions in the first model and use a time parameter Δt to screen out those statistics out of time range; hence, we can analyze real-time performance and offer the team some suggestions on improving efficiency. If the team has an opponent, we can also use the network to analyze the opponent's performance and put forward some strategies to make our team more competitive.

This model can be used to analyze the team performance and individual performance most of the time. To generalize the model, we need to capture more aspects of teamwork such as the work procedure before a successful teamwork (the series of pass before successful shot or goal in soccer game). These aspects can provide us with a more direct way of judgement, which will improve our model and make it more general.

6 Model Analysis

6.1 Sensitivity Analysis

Our model contains several parameters. We determine some of the parameters through Analytic Hierarchy Process (AHP), some of them by knowledge in the literature, and other methods. In this section, we would like to produce a sensitivity analysis to show whether our model is sensitive to different values of parameters. We will choose the impact of an important parameters Δt in our model as representative.

Δt is a RTPN parameters with the meaning of the time period of the statistics. We set Δt to 10 minutes in Section 5.3. Figure 13 shows the connection between different Δt and real-time clustering coefficient of player M1 in match 1.

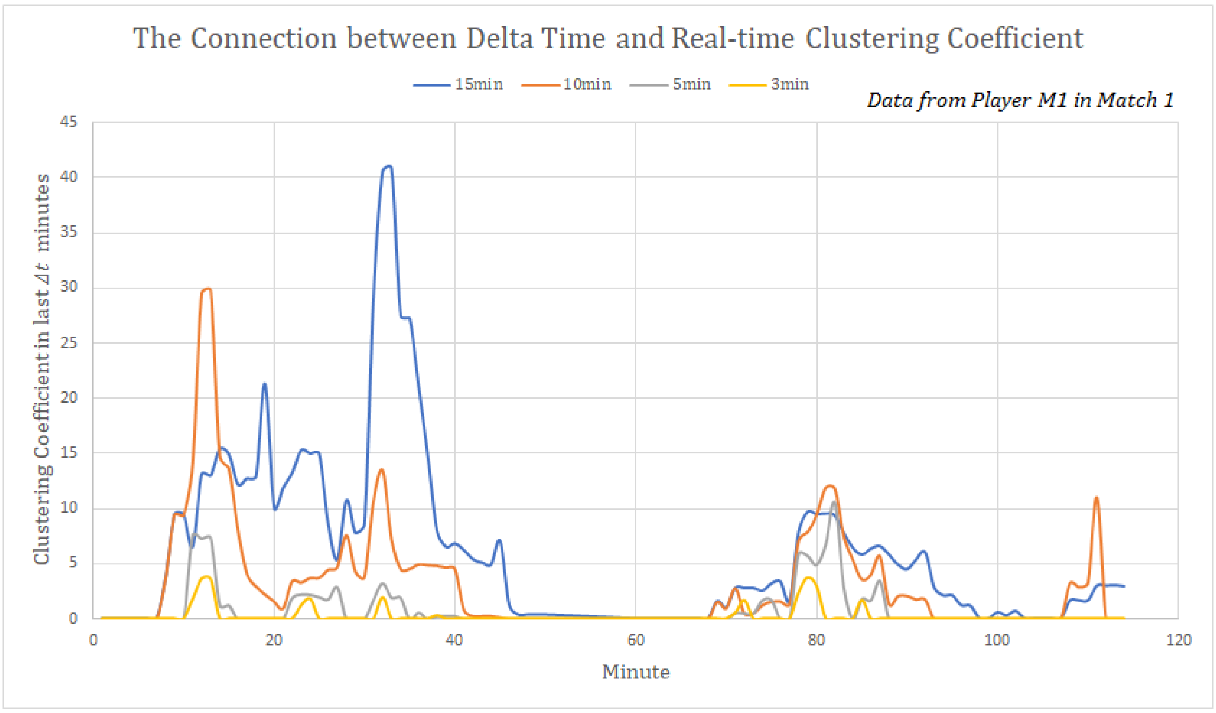


Figure 13: The real-time clustering coefficient of player M1 in match 1 with different Δt

From the figure above we find that for all Δt, the curves reach their peaks in almost the same time. But when the time period Δt is too short (e.g. $\Delta t = 5\ minutes$), some of the peaks disappear because the statistics are not enough for our RTPN to analyze player's performance. When the time period Δt is too long (e.g. $\Delta t = 15\ minutes$), some peaks begin to combine with each other and the previous statistics and have a huge influence in current statistics and analysis, which makes our analysis not precise enough. In conclusion, our RTPN model is sensitive to its parameter Δt, and the setting of Δt to 10 minutes is reasonable.

6.2 Strengths and Weaknesses

6.2.1 Strengths

1. Our model is based on scientific methods like **K-Means**,**AHP** and **the min-cut model**, which make our model more reliable.
2. We modify the traditional clustering coefficient to fit our model and creatively integrate the individuality of players into their teamwork.
3. Various forms of figures are displayed to visualize our model.

6.2.2 Weaknesses

1. When we calculate the weight of edges in the Passing Network, we take no account of the contribution it makes,such as leading to a goal.
2. For lack of opponents' data, our structural strategies before a match may not be flexible enough.

7 Conclusion

Our paper analyzes the previous match data of the Huskies to form a comprehensive awareness of both teamwork and personal features. Then we design the real-time passing network - player combining (PnP) model consisting of three sub-models, namely Passing Network, Player Model and Real-Time Passing Network.

With the help of our model, we redefine the clustering coefficient and figure out its relation with team performance. Network patterns like partial cooperation between two or three players and structural strategies are also concerned.

We really hope our model and recommendation can be useful for both Huskies and teamwork in our society.

References

[1] "How data(and some breathtaking soccer) brought liverpool to the cusp of glory," New York Times Magazine, 2019. [Online]. Available: https://www.nytimes.com/2019/05/22/magazine/soccer-data-liverpool.html

[2] V. A. BARRAT A, BARTHELEMY M, "The architecture of complex weighted networks:measurements and models," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101(11), pp. 37–47, 2003.

[3] M. R. W. T. Page L, Brin S, "The pagerank citation ranking: Bringing order to the web," *Stanford InfoLab*, 1999.

[4] T. L. Saaty, "How to make a decision: The analytic hierarchy process," *European Journal of Operational Research*, vol. 48, pp. 9–26, 1990.

[5] "Max-flow min-cut theorem," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Max-flow_min-cut_theorem

[6] "Dinic algorithm," Wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Dinic%27s_algorithm

Appendices

Appendix A Edge Weight and Pass Clustering Coefficient Calculation

```
# Create Pass Network and Calculate Clustering Coefficient for each player.
# Note: csvreader is our custom library to read csv file.
import csvreader
import csv
import math
import numpy as np
import matplotlib.pyplot as plt


def add_coordinates(loc, s, x, y):
    if loc.get(s) is None:
        loc[s] = [x, y, 1]
    else:
        lst = loc[s]
        loc[s] = [lst[0] + x, lst[1] + y, lst[2] + 1]
    return None


def add_counts(dicts, s, add):
    if dicts.get(s) is None:
        dicts[s] = [add, 1]
    else:
        dicts[s] = [dicts[s][0] + add, dicts[s][1] + 1]
    return None


def get_counts(dicts, s):
    if dicts.get(s) is None:
        return 0
    else:
        return dicts[s][0] / dicts[s][1]


def euclid_distance(x, y, xx, yy):
    return math.sqrt((x - xx) ** 2 + (y - yy) ** 2)


def euclid_distance_loc(a, b):
    return euclid_distance(a[0], a[1], b[0], b[1])


def main(write_match):
    full_events = csvreader.csv_reader_without_headers('../data/fullevents.csv')
    passing_events = csvreader.csv_reader_without_headers('../data/passingevents.csv')
    # Only consider our team: Huskies.
    write_team = 'Huskies'
    location = {}
    substitution_list = []
    name_list = []
    # Count substitutions and average coordinates of players.
    for item in full_events:
        if int(item[0]) is not write_match:
            continue
        if item[1].find(write_team) is -1:
            continue
        name = (item[2].split('_'))[1]
        if name not in name_list:
            name_list.append(name)
        if item[6].find('Pass') is not -1:
            add_coordinates(location, name, float(item[8]), float(item[9]))
            if item[3] is not '' and item[3] != item[2]:
                add_coordinates(location, (item[3].split('_'))[1],
                                float(item[10]), float(item[11]))
        elif item[6].find('Save') is not -1 or item[6].find('Goalkeeper') is not -1:
            add_coordinates(location, name, 0, 50)
        elif item[6].find('Substitution') is not -1:
            substitution_list.append((item[3].split('_'))[1])
        elif item[6].find('Duel') is not -1 or item[6].find('Others') is not -1 or \
             item[6].find('Foul') is not -1 or item[6].find('Offside') is not -1 or \
             item[6].find('Shot') is not -1:
            add_coordinates(location, name, float(item[8]), float(item[9]))
    # Compute the player's average location.
    for key in location:
        loc = location[key]
        assert loc[2] is not 0
        location[key] = [loc[0] / loc[2], loc[1] / loc[2]]
    # Compute the average coordinate distance between players.
    dC = {}
    for name1 in name_list:
        for name2 in name_list:
            if name1 == name2:
                continue
            key = name1 + '.' + name2
            dC[key] = euclid_distance_loc(location[name1], location[name2])
    # Count the times of each pass and passing times distance
    head_pass = {}
    hand_pass = {}
    high_pass = {}
    simple_pass = {}
    launch = {}
    cross = {}
    smart_pass = {}
    dT = {}
    for name1 in name_list:
        for name2 in name_list:
            if name1 == name2:
                continue
            key = name1 + '.' + name2
            dT[key] = 0
    for item in passing_events:
        if int(item[0]) is not write_match:
            continue
        if item[1].find(write_team) is -1:
            continue
        name1 = (item[2].split('_'))[1]
        name2 = (item[3].split('_'))[1]
        if name1 == name2:
            continue
        key = name1 + '.' + name2
        dT[key] += 1
        # Use euclid distance.
        dis = euclid_distance(float(item[7]), float(item[8]), \
                              float(item[9]), float(item[10]))
        if item[6].find('Head') != -1:  # Head pass
            add_counts(head_pass, key, dis)
        elif item[6].find('Hand') != -1:  # Hand pass
            add_counts(hand_pass, key, dis)
        elif item[6].find('High') != -1:  # High pass
            add_counts(high_pass, key, dis)
        elif item[6].find('Simple') != -1:  # Simple pass
            add_counts(simple_pass, key, dis)
        elif item[6].find('Launch') != -1:  # Launch
            add_counts(launch, key, dis)
        elif item[6].find('Cross') != -1:  # Cross
            add_counts(cross, key, dis)
        elif item[6].find('Smart') != -1:  # Smart pass
            add_counts(smart_pass, key, dis)
    # Compute the average passing distance.
    dP = {}
    kdP = {'Head pass': 0.8, 'Hand pass': 0.6, 'High pass': 1.3, 'Simple pass': 1.0,
           'Launch': 0.7, 'Cross': 1.2, 'Smart pass': 1.1}
    kdP_total = kdP['Head pass'] + kdP['Hand pass'] + kdP['High pass'] + \
                kdP['Simple pass'] + kdP['Cross'] + kdP['Smart pass'] + kdP['Launch']
    for name1 in name_list:
        for name2 in name_list:
            if name1 == name2:
                continue
            key = name1 + '.' + name2
            score = get_counts(head_pass, key) * kdP['Head pass'] \
                    + get_counts(hand_pass, key) * kdP['Hand pass'] \
                    + get_counts(high_pass, key) * kdP['High pass'] \
                    + get_counts(simple_pass, key) * kdP['Simple pass'] \
                    + get_counts(launch, key) * kdP['Launch'] \
                    + get_counts(cross, key) * kdP['Cross'] \
                    + get_counts(smart_pass, key) * kdP['Smart pass']
            score = score / kdP_total
            dP[key] = score
    # Compute dT and ready for standardize
    maxdC = 0
    maxdP = 0
    for name1 in name_list:
        for name2 in name_list:
            if name1 == name2:
                continue
            key = name1 + '.' + name2
            if dT[key] == 0:
                continue
            dT[key] = 1 / dT[key]
            maxdC = max(maxdC, dC[key])
```

```
                maxdP = max(maxdP, dP[key])
        # Standardize & Compute weight of each edge.
        w = {}
        for name1 in name_list:
            for name2 in name_list:
                if name1 == name2:
                    continue
                if name1 in substitution_list or name2 in substitution_list:
                    continue
                key = name1 + '.' + name2
                if dT[key] == 0:
                    continue
                # Standardize.
                dC[key] = dC[key] / maxdC
                dP[key] = dP[key] / maxdP
                # Calculate weight of each edge.
                w[key] = 1 / (0.7 * dT[key] + 0.2 * dC[key] + 0.1 * dP[key])
                w[key] = math.exp(w[key])
        # Write weight result to an file.
        headers = ['From', 'To', 'w']
        file_name = '../data/networkdata/match' + str(write_match) + '_w.csv'
        with open(file_name, 'w', newline='') as csv_file:
            csv_writer = csv.writer(csv_file, dialect='excel')
            csv_writer.writerow(headers)
            for key in w:
                [name1, name2] = key.split('.')
                csv_writer.writerow([name1, name2, w[key]])
        # Calculate Clustering Coefficient of every player.
        a = {}
        k = {}
        for name1 in name_list:
            count = 0
            for name2 in name_list:
                key = name1 + '.' + name2
                if w.get(key) is None:
                    a[key] = 0
                else:
                    a[key] = 1
                    count += 1
            k[name1] = count
        c_count = 0
        c_num = 0
        c = {}
        for name in name_list:
            sum = 0
            for name2 in name_list:
                for name3 in name_list:
                    key1 = name2 + '.' + name
                    key2 = name + '.' + name3
                    key3 = name2 + '.' + name3
                    if a[key1] == 1 and a[key2] == 1 and a[key3] == 1:
                        sum = sum + (w[key1] + w[key2]) / 2
            if k[name] != 0 and k[name] != 1:
                c[name] = sum / k[name] / (k[name] - 1)
                c_count += c[name]
                c_num += 1
            else:
                c[name] = 0
        # Write result to an file.
        headers = ['Name', 'Clustering Coefficient']
        file_name = '../data/networkdata/match' + str(write_match) + '_c.csv'
        with open(file_name, 'w', newline='') as csv_file:
            csv_writer = csv.writer(csv_file, dialect='excel')
            csv_writer.writerow(headers)
            for key in c:
                if c[key] is not 0:
                    csv_writer.writerow([key, c[key]])


if __name__ == '__main__':
    # Compute the data of match 1.
    main(1)
```

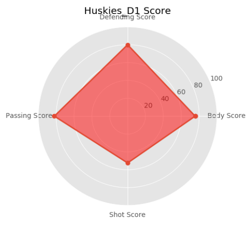
Appendix B PageRank Algorithm

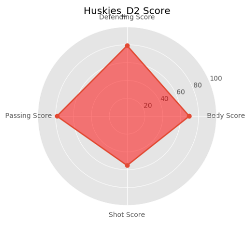
```
# Calculating Page rank.
class Graph():
    def __init__(self):
        self.linked_node_map = {}
        self.PR_map = {}

    def add_node(self, node_id):
        if node_id not in self.linked_node_map:
            self.linked_node_map[node_id] = []
            self.PR_map[node_id] = 0

    def add_link(self, node1, node2, v):
        if node1 not in self.linked_node_map:
            self.add_node(node1)
        if node2 not in self.linked_node_map:
            self.add_node(node2)
        # When Inserting links, weights are already divided by sum.
        self.linked_node_map[node2].append([node1, v])

    # Compute Page Rank
    def get_PR(self, epoch_num=50, d=0.95):
        for i in range(epoch_num):
            for node in self.PR_map:
                self.PR_map[node] = (1 - d) + d * sum(
                    [self.PR_map[temp_node[0]] * temp_node[1]
                     for temp_node in self.linked_node_map[node]])
        return self.PR_map
```

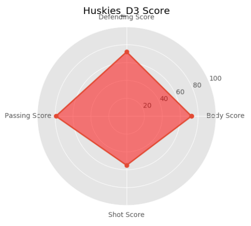
Appendix C K-Means Clustering Algorithm

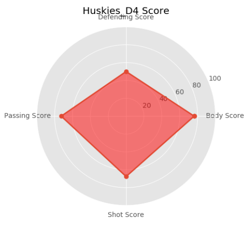
```
# KMeans Classification
# Note: csvreader is our custom library to read csv file.
import csvreader
import random


def kmeans_step(K, c, bel, center):
    loss = 0
    # Calculate current total loss.
    for i, item in enumerate(c):
        # Choose the nearest center.
        min_loss, pos = 1e9, 0
        for j in range(0, K):
            dis = (center[j] - item[1]) ** 2
            if dis < min_loss:
                min_loss, pos = dis, j
        bel[i] = pos
        loss += min_loss
    num = []
    for i in range(0, K):
        center[i] = 0
        num.append(0)
    # Calculate new center.
    for i in range(0, len(c)):
        center[bel[i]] += c[i][1]
        num[bel[i]] += 1
    for i in range(0, K):
        center[i] /= num[i]
    return loss


def kmeans(c, write_match, K):
    # Randomly choose K centers.
    rd_list = []
    for i in range(0, K):
        x = random.randint(0, len(c) - 1)
        while x in rd_list:
            x = random.randint(0, len(c) - 1)
        rd_list.append(x)
    bel = []
    for item in c:
        bel.append(0)
    center = []
    for i in rd_list:
        center.append(c[i][1])
    ori_loss = kmeans_step(K, c, bel, center)
    while True:
        loss = kmeans_step(K, c, bel, center)
        # if loss is stable, then classification is finished.
        if abs(loss - ori_loss) < 1e-3:
            break
        ori_loss = loss
    return center, bel


def main(write_match):
    c_data = csvreader.csv_reader_without_headers('../data/networkdata/match'
                                                  + str(write_match) + '_c.csv')
    # c means clustering coefficient data.
    # c[0] is player name, c[1] is the clustering coefficient value.
    c = []
    for item in c_data:
        c.append([item[0], float(item[1])])
    # First divide into 2 groups: Group A and others.
    center, bel = kmeans(c, write_match, 2)
    # Make center[1] group A.
    if center[0] > center[1]:
        center[0], center[1] = center[1], center[0]
        for i, item in enumerate(bel):
            if item == 1:
                bel[i] = 0
            else:
                bel[i] = 1
    # Copy other item to d[].
    d = []
    for i, item in enumerate(c):
        if bel[i] == 0:
            d.append(item)
    # Divide others into 2 groups: Group B and Group C.
    center2, bel2 = kmeans(d, write_match, 2)
    # Make center2[0] Group C and center2[1] Group B.
    if center2[0] > center2[1]:
        center2[0], center2[1] = center2[1], center2[0]
    # Return every group's center.
    return [center2[0], center2[1], center[1]]

if __name__ == '__main__':
    # Analyze the data of match 1.
    print(main(1))
```

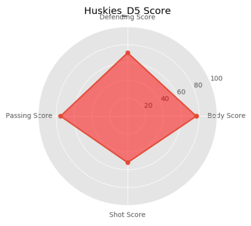
Appendix D Scores of all players (The Overall and Detailed Version)

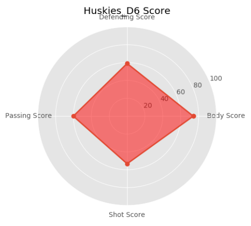


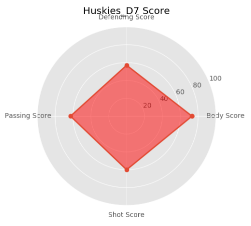


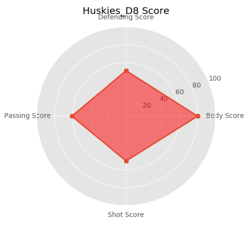


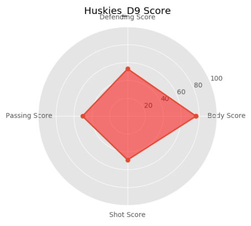


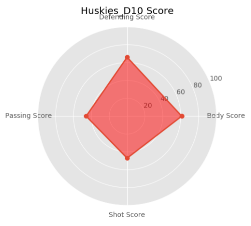


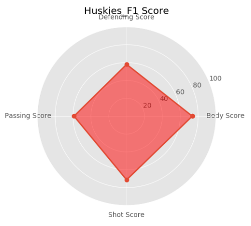


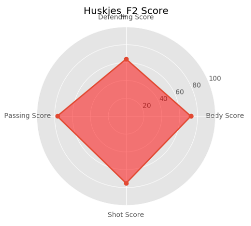


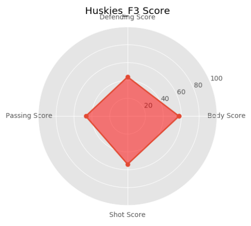


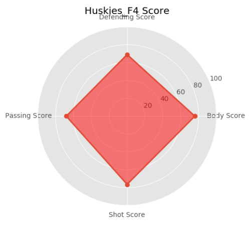


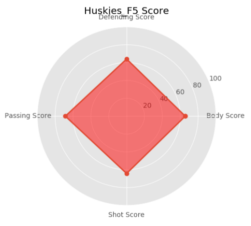


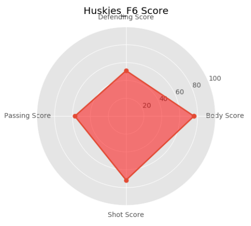


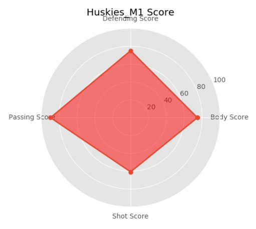


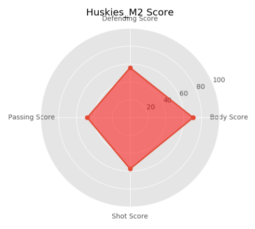


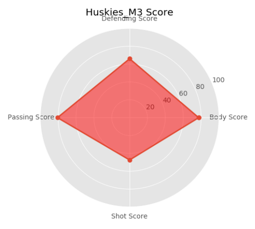


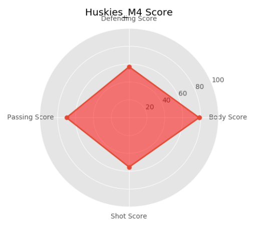


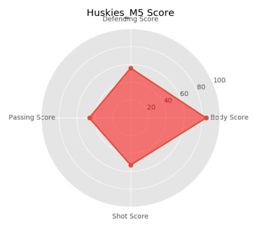


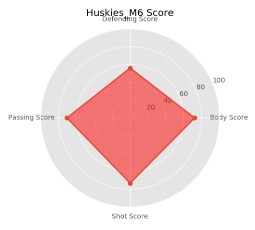

Huskies_M1 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

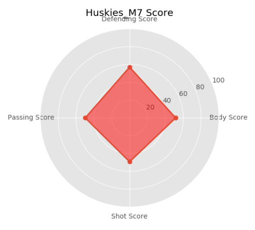

Huskies_M2 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

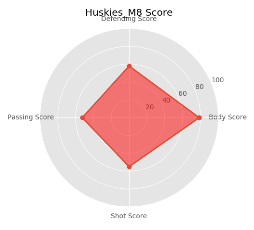

Huskies_M3 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

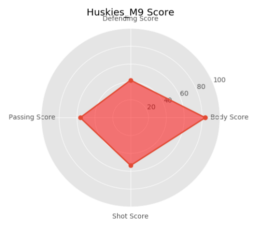

Huskies_M4 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

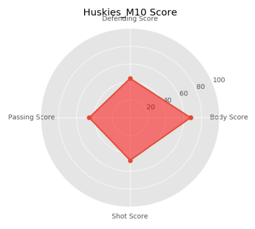

Huskies_M5 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

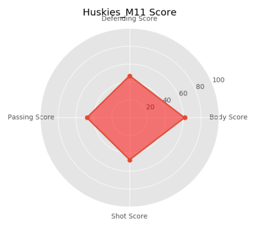

Huskies_M6 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

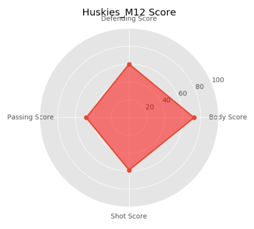

Huskies_M7 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

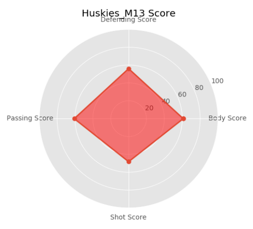

Huskies_M8 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

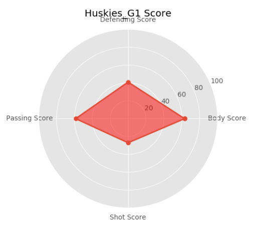

Huskies_M9 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

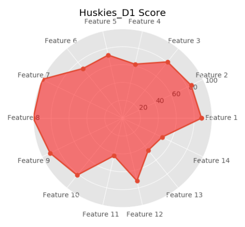

Huskies_M10 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

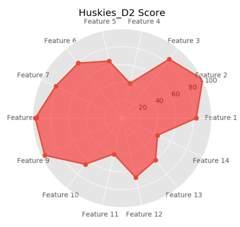

Huskies_M11 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

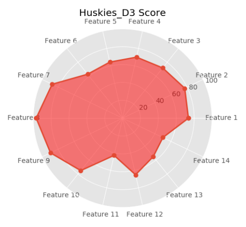

Huskies_M12 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

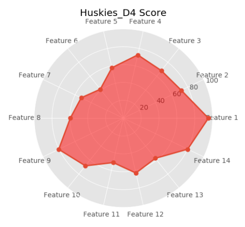

Huskies_M13 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

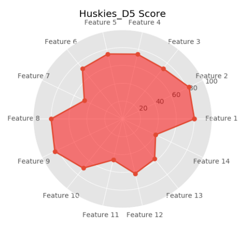

Huskies_G1 Score
Defending Score
Passing Score
Body Score
Shot Score
20
40
60
80
100

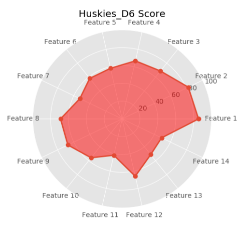

Huskies_D1 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

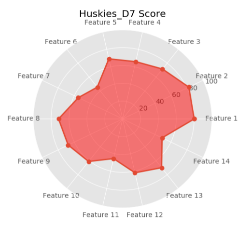

Huskies_D2 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

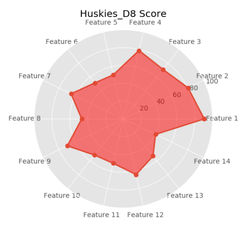

Huskies_D3 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

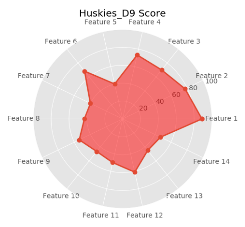

Huskies_D4 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

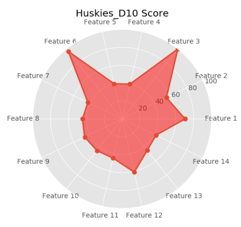

Huskies_D5 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

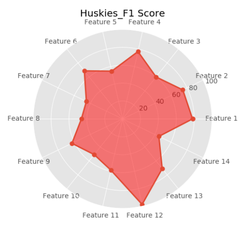

Huskies_D6 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

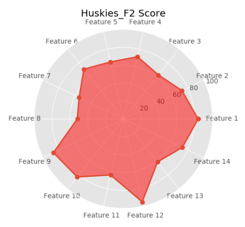

Huskies_D7 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

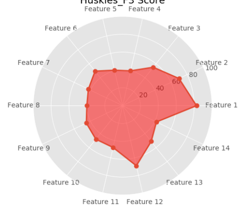

Huskies_D8 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

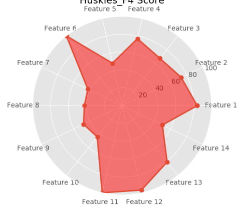

Huskies_D9 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

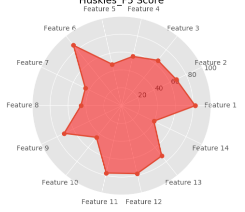

Huskies_D10 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

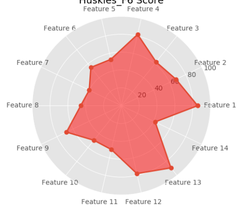

Huskies_F1 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

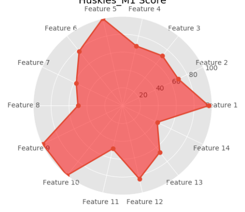

Huskies_F2 Score
Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14
20
40
60
80
100

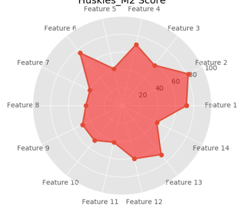
Huskies_F3 Score


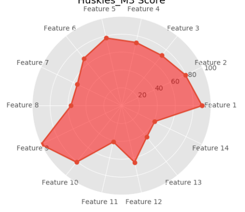
Huskies_F4 Score


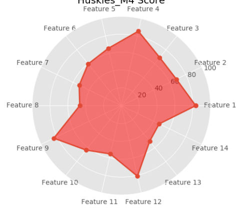
Huskies_F5 Score


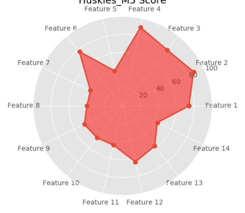
Huskies_F6 Score


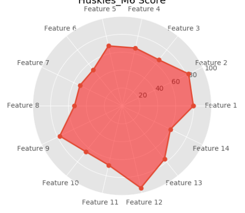
Huskies_M1 Score


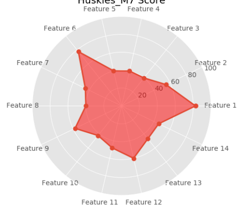
Huskies_M2 Score


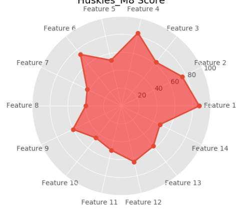
Huskies_M3 Score


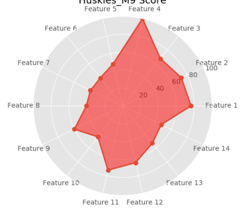
Huskies_M4 Score


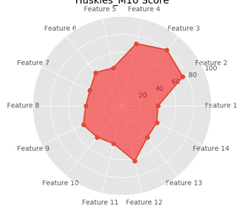
Huskies_M5 Score


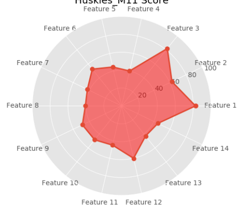
Huskies_M6 Score


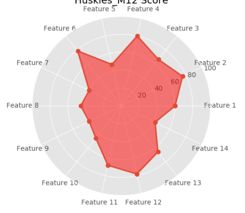
Huskies_M7 Score


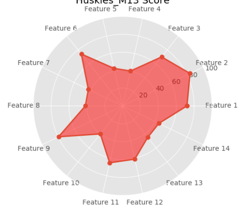
Huskies_M8 Score


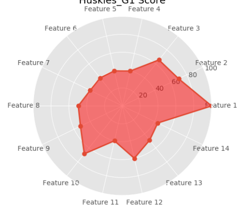
Huskies_M9 Score


Huskies_M10 Score


Huskies_M11 Score


Huskies_M12 Score


Huskies_M13 Score


Huskies_G1 Score

Feature 1
Feature 2
Feature 3
Feature 4
Feature 5
Feature 6
Feature 7
Feature 8
Feature 9
Feature 10
Feature 11
Feature 12
Feature 13
Feature 14