UCLouvain

LOUVAIN
School of Management

A SHORT STUDY OF PAIRWISE COMPARISON METHODS APPLIED TO THE RANKING OF SPORT TEAMS

Author: Brent Vansprengel
Promotor: Prof. Marco Saerens

Academic Year: 2024-2025

Master thesis submitted in order to be awarded the degree of:
Master in Management [60]

UCLouvain

LOUVAIN
School of Management

A SHORT STUDY OF PAIRWISE COMPARISON METHODS APPLIED TO THE RANKING OF SPORT TEAMS

Author: Brent Vansprengel
Promotor: Prof. Marco Saerens

Academic Year: 2024-2025

Master thesis submitted in order to be awarded the degree of:
Master in Management [60]

Study of pairwise comparison methods and their functionality in ranking mechanisms

Companies and organisations rely on managers and their decision making skills to achieve organisational objectives with the available resources. Whether it concerns the HR affairs or the logistical chain of the company, pairwise comparison methods and ranking mechanisms, as explored in this paper, offer an interesting framework for deriving meaningful insights across various management domains.

From a Human Resources perspective, ranking mechanisms are instrumental in performance evaluations and talent identification. Organisations will too often grapple with subjective judgements or, on the other end, rely too heavily on over-quantified metrics. Given the right framework and the proper understanding of this framework, these organisations can asses personnel performance based on a representative ranking. This in turn allows for managers to provide more nuanced guidance towards their employees and to make better informed decisions.

In marketing, establishing targeted campaigns relies on the understanding of consumer preferences. Pairwise comparisons in all possible forms provide more insight to these preferences. Managers who know how to interpret the outcome of these comparisons can make informed decisions towards a better resource allocation. Furthermore, pairwise comparisons in the form of reviews or surveys provide rankings that can aid product development and refinement

Contract allocation and supplier selection present another compelling application. Management often has to evaluate competing proposals where countless factors should be taken into account. A good understanding of graph theory could help set up a ranking where direct links are not evident. Such an approach improves decision accuracy but also ensures agreements are based on objective and defensible criteria.

While using the football pitch as a starting point, this paper aims to illustrate the broader applicability of pairwise comparison methods and ranking mechanisms, and to underscore the value of understanding their use. After a theoretical chapter, this paper studies four ranking mechanisms more in depth: The points based ranking, the Elo rating, the Schulze method and the Floyd-Warshall algorithm. Interpreting match outcomes as pairwise preferences between teams, the different ranking mechanisms are evaluated on their ability to reflect team performances and predict match outcomes. A cross-validation technique is used to assess predictive accuracy and the mean squared error between an expected outcome and an actual outcome.

If managers are willing to investigate pairwise comparison methods and ranking mechanisms, organisations can benefit from transparent and deliberate decision making in a broad range of domains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ranking mechanisms are embedded around us. They are fundamental tools, used in decision-making processes across all sorts of domains. Whether you are evaluating the performance of your favourite football team, considering which candidate to support during an election, or deciding which one of your employees to appoint as your successor, rankings provide a quantified overview in an attempt to make informed choices. However, the mechanism behind a ranking can vary significantly. Each with unique strengths, weaknesses and contextual applications.

This paper studies pairwise comparison methods and ranking mechanisms. It delves into the theoretical background before applying these principles on a practical example. First of all, the theoretical study introduces key concepts in voting systems. The fundamentals of these voting systems rely on pairwise comparisons, which means they can provide valuable insights in establishing rankings. Next, graph theory is introduced through the Floyd-Warshall algorithm, followed by an introduction of the Elo rating. To conclude the theory, the randomised shortest path theory is introduced. But, regardless of its potential, this theory will not return later on as it would drive us too far in our practical implementation.

From this theoretical background, the paper continues to the practical implementation of four ranking mechanisms on a real-world scenario: The points based ranking, the Elo ranking, the Schulze ranking and the Floyd-Warshall ranking. Seasons '20-'21 and '21-'22 of the Belgian Jupiler Pro League will serve as the two primary case studies. Interpreting match outcomes as pairwise preferences between teams, the different ranking mechanisms are evaluated on their ability to reflect team performances and predict match outcomes. A cross-validation technique is used to assess predictive accuracy and the mean squared error between an expected outcome and an actual outcome.

While there is plenty to learn in this paper about the theoretical foundations and practical nuances of ranking mechanisms, readers should not expect exhaustive research of all ranking methodologies beyond those explicitly discussed. The analysis of different mechanisms, based on the data from two football seasons, is intended to provide a proper understanding of the strengths and limitations of each system. However, this limited dataset precludes definitive conclusions or excessive extrapolations. Instead it should be an incentive for further research to validate findings on a larger scale.

This paper aims to provide valuable and comprehensive insights about the question: How do different pairwise comparison methods fare in a dynamic, competitive, real-world environment and what can we learn from their practical implementation?

Chapter 2

Theoretical background of voting systems and ranking mechanisms

This chapter introduces a theoretical background to the main principles used throughout this paper. It covers several voting systems and voting criteria, which are based on pairwise comparisons. Furthermore it introduces graph theory through the Floyd-Warshall algorithm and shortly touches on the randomised shortest path problem. While this chapter seeks to provide most of the required knowledge to understand this paper, numerous references are made to further studies where readers can delve even deeper into the concerned topics.

2.1 Voting systems

Voting systems have been subjected to studies for centuries. They provide a structure to translate individual preferences into a collective decision, ensuring that the opinion of a group or society is reflected in the outcome. A specific set of preferences can lead to a different outcome, depending on the properties of a voting system. This only emphasizes the importance of understanding the mechanisms and their limitations.

Ranking mechanisms have a lot in common with voting systems. By exploring different voting systems and voting criteria, rankings can be analysed and developed with a better understanding.

2.1.1 Plurality voting

In plurality voting, voters have to express their first preference. The option with the most preference votes wins. Take an election with 3 options and 20 voters as a simple example:

- A is chosen as most preferred 9 times,
- B is chosen as most preferred 7 times,
- C is chosen as most preferred 4 times,

Option A wins as it has the most votes, without a need for an absolute majority. Plurality voting is simple, quick to count and encourages a decisive outcome. However, this decisiveness can result in a winner who is not backed by the majority of voters, as only a first choice is taken into consideration. This leaves the system open to strategic voting where a less preferred but more viable candidate could be chosen.

Plurality voting is common in political elections, including in the US and the UK. These countries are a prime example of Duverger's law [7], which states that plurality voting will often lead to a two-party system. This is due to the tendency to marginalize smaller parties, which forces people to vote strategically.

2.1.2 Approval voting

In approval voting, people can select any number of options on the ballot, giving them their approval. This system shows no individual ranking, but will favour the broadly acceptable options. Furthermore, it is believed that approval voting entices voting participation [2] and avoids a centre squeeze, like we see nowadays in numerous national elections [3]. Using the same example as before, voters now have the chance to approve multiple options:

- A has 12 approvals
- B has 14 approvals
- C has 7 approvals

It's known that most voters prefer option A to option B, as was stated during the plurality voting. Nonetheless, we see that option B wins, as it has the most approvals. This is a violation of the Condorcet criterion (sec. 2.2.1).

2.1.3 Borda count

The Borda count is a ranked voting system that allows people to rank all options in order of their preference. They can assign points based on this individual pairwise preference. In general, the most preferred option receives the maximum points, which is equal to the amount of options minus one. The second-ranked receives one less than the maximum and so on. The candidate with the highest count wins. Using the same example as before, voters now have the chance to rank all options:

- Ranking A $>$ B $>$ C was given 6 times
- Ranking A $>$ C $>$ B was given 3 times
- Ranking B $>$ A $>$ C was given 2 times
- Ranking B $>$ C $>$ A was given 5 times
- Ranking C $>$ A $>$ B was given 3 times
- Ranking C $>$ B $>$ A was given 1 time

These results render the following outcome: Option A receives nine times two points, five times two points and six times no points, resulting in a score of 23. Similarly, B receives 21 points and C 16 points. Once again, A is the winner according to the Borda count.

Like the approval voting system, voters are encouraged to consider all candidates and thus the result will reflect a broader preference. Unfortunately, also like an approval voting, it fails to comply with the Condorcet criterion. In the 18^{th} century, Nicolas de Condorcet did extensive research and demonstrated that the Borda count violates the principle of "independence of irrelevant alternatives". This axiom states that a choice between option A and B should not depend on the quality of option C, when option C is entirely unrelated [14].

Most American baseball fans will not be aware of this fact, but the "Major League Baseball Most Valuable Player", or "MVP award", is elected through a Borda count.

2.1.4 Schulze method

The Schulze method, also known as the beatpath method, is a ranked voting system that aims to find a ranking based on pairwise comparisons between all options. It complies with the Condorcet criterion (sec. 2.2.1), which means that if a Condorcet winner exists, that option will win [16].

First, a pairwise comparison matrix will be established. $P_{X,Y}$ equals the strength of the preference for X over Y. Next, the strongest path has to be found for each pair of options. A path can be any sequence of pairwise comparisons connecting two options. Its strength is determined by the weakest link. Finally, a ranking is established by comparing the strongest paths between options. Option X is ranked higher than option Y if the strongest path from X to Y is stronger than the reverse path.

Looking back at the previous example (sec. 2.1.3), the following pairwise preference matrix can be established (fig. 2.1).

	A	B	C
A	/	6 + 3 + 3 = 12	6 + 3 + 2 = 11
B	2 + 5 + 1 = 8	/	6 + 2 + 5 = 13
C	5 + 3 + 1 = 9	3 + 3 + 1 = 7	/

Figure 2.1: Pairwise preference matrix

In order to find the strongest path, the matrix is transformed to create a preference graph (fig. 2.2). The options are represented through the nodes and the margin of preference equals the weight of the link. In this example, the preference for A over B, $P_{A,B}$, equals 12, whilst $P_{B,A}$ equals 8. Link A $\rightarrow$ B is added to the graph with a weight of 4.

	A	B	C
A	/	12 - 8 = 4	11 - 9 = 2
B	8 < 12 → -∞	/	13 - 7 = 6
C	9 < 11 → -∞	7 < 13 → -∞	/

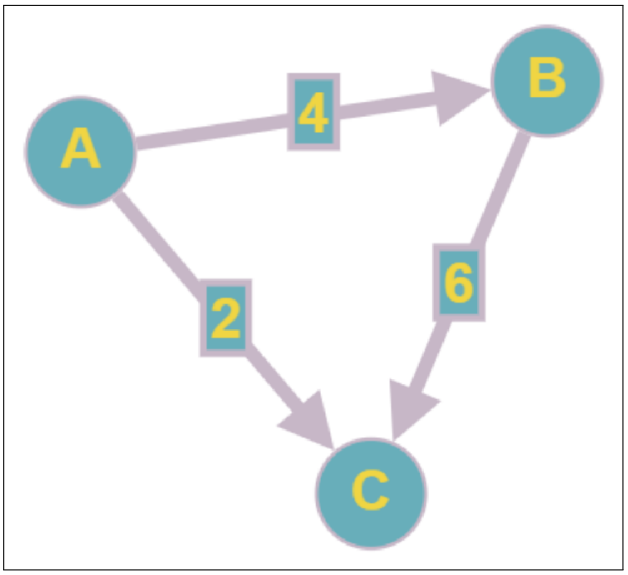

Figure 2.2: Preference graph

Now looking for the strongest path between A and B, the direct path is identified with a strength of 4. But when considering the strongest between A and C, the indirect path from A to C through B, with a minimal strength of 4, triumphs the direct path which only has strength of 2. The strength of a path is the strength of its weakest link.

In this example, there remains a clear overview due to the few options, which means finding the strongest paths can be done manually. However, this becomes far too complex in real world applications. An algorithm (A.2) is used to go through all pairs of nodes and compares all possible paths.

This algorithm, after using all nodes to compare each pairing, returns a matrix with all the strongest paths between any pair of options (fig. 2.3).

	A	B	C
A	/	4	4
B	- ∞	/	6
C	- ∞	- ∞	/

Figure 2.3: Strongest path matrix

The final step of the Schulze method is to determine, for each option, the number of options it can triumph, based on the strongest path matrix. This results in a Dominance score for each option, creating a ranking. In this example, it's A → B → C.

2.2 Voting criteria

Objective voting criteria are essential when working with all kinds of voting systems. These criteria help identify flaws, benefit transparency and consistency, and aid in the search for fairer methods. They also help the general public understand the trade-offs between systems and thus promote informed choices in ranking and decision making.

The criteria can be categorised as absolute criteria, relative criteria, ballot counting criteria and strategy criteria. The most relevant ones for this paper will be discussed below. A more in-depth study can be found the work of D.R. Woodall, Properties of Preferential Election Rules [22].

	Majority winner	Condorcet winner	Smith	IIA	Monotone	Later no harm	Later no help	No favourite betrayal
Plurality voting	Pass	Fail	Fail	Fail	Pass	Pass	Pass	Fail
Approval voting	Pass	Fail	Fail	Pass	Pass	Fail	Pass	Pass
Borda count	Fail	Fail	Fail	Fail	Pass	Fail	Pass	Fail
Schulze Method	Pass	Pass	Pass	Fail	Pass	Fail	Fail	Fail

Figure 2.4: Prominent voting criteria [19]

2.2.1 Absolute voting criteria

Majority criterion

The majority criterion is a fundamental principle in voting and ranking systems that ensures the victory of a majority preferred option, if one exists. It states that if option A is ranked first by a majority, A will always win. The importance of this criterion was recently very clear during the US presidential election of 2024. Although the president elect does not need a majority approval from the entire population, satisfying the criterion makes the result more credible [17].

Point based voting mechanisms, like the Borda count, can fail this criterion. They might select an option with a broad but weaker support over one with a majority first-place support.

Condorcet criterion

The Condorcet criterion states that if an option A exists that is preferred over each alternative option in a pairwise comparison, this option A should be ranked first. It emphasises the correctness in a pairwise comparison, prioritizing options with the strongest overall support [4].

There are common voting mechanisms, like plurality voting, that don't meet this criterion. There are multiple examples in recent history where the Condorcet criterion was violated. One of the most prominent examples is the 2000 US presidential election in Florida, where Al-Gore lost the election to G. Bush, despite being the preferred candidate [5].

Smith criterion

The Smith criterion requires that the top ranked option belongs to the Smith set. This Smith set can be defined as the smallest group of options that defeat all options outside the set in pairwise comparisons. The Smith set is collectively stronger than any candidate not in this set.

It can be seen as an extension of the Condorcet criterion. The Condorcet winner will always be part of the Smith set as it defeats every other option in pairwise comparison. However, the Smith criterion can even be applied when no Condorcet winner exists [21].

2.2.2 Relative voting criteria

Independence of irrelevant alternatives criterion, IIA

The IIA states that if option A is chosen over option B, introducing option C will not result in option B being chosen over A. The same goes for removing option C from the original choice.

Although the IIA criterion may seem fair, it can also be considered as an undesirable or unrealistic criterion. Taking the example of Nobel prize winner Amartya Kumar Sen: A person considering whether to take an apple out of a basket without being greedy. If the only two options available are "take the apple" or "don't take the apple", this person may conclude that there is only one apple left and so refrain from taking the last apple as they don't want to be greedy. However, if a third option "take another apple" were available, that would provide context that there are more apples in the basket, and they would then be free to take the first apple [20].

Monotonicity criterion

The monotonicity criterion states that if option A improves in individual preferences by receiving higher rankings, A should never receive a lower global ranking. Failing this criterion could lead to an option being ranked lower despite gaining support, resulting in unfair outcomes.

Certain voting systems, especially systems that involve elimination rounds, violate this criterion. One of these systems is the Instant Runoff Voting, IRV, which displayed this flaw during the 2009 Burlington mayoral election. In 2009, Bob Kiss won the re-election, defeating Kurt Wringt in the final round. However, Kiss would have been eliminated earlier in the counting process had he received additional first-choice votes from supporters of another candidate [1].

2.2.3 Strategy Criteria

Later-no-harm criterion

The later-no-harm criterion states that ranking option A first and B second would not harm A's chances. It encourages voters to express their honest range of preferences without a fear that supporting additional options could affect their primary option.

Voting systems like the Borda count, where points are assigned as per individual rankings, do not comply with this criterion. If a close race is expected between option A and option B, one could rank option C, D, E, etc. above option B in order to harm the score of B and create an advantage for option A.

Later-no-help criterion

The later-no-help criterion states that ranking a less preferred option B higher cannot cause a more preferred option A to win. This criterion is incompatible with the Condorcet criterion, which means the Schulze method is susceptible for this tactical voting strategy. When done on a large scale in specific conditions, the real Condorcet preferred option could lose to a manipulated alternative.

No favourite betrayal criterion

The no favourite betrayal criterion states that a voter preferring option A, should never have to rank option B higher than option A in order to achieve a better outcome for option A.

2.3 Graph theory

Graph theory is an branch within mathematics and computer science that studies mathematical structures representing pairwise relationships (edges) between objects (nodes). It forms an indispensable tool in network analysis, logistical problems or optimisation problems. An in-depth study about graph theory, is the work of F. Fouss et al., Algorithms and Models for Network Data and Link Analysis [11].

2.3.1 Floyd-Warshall algorithm

The computational backbone of the Schulze method is called the Floyd-Warshall algorithm [11]. While the Schulze method is a ranked voting system that is looking for the strongest paths between different options, the Floyd-Warshall algorithm is solely interested in the shortest paths, with a broad applicability beyond voting systems [16]. Instead of preferences of A over B, we will address the cost of going from node i to j in the graph.

$$\Delta_{ik} \triangleq \min_{\wp \in P_{ik}} \{\tilde{c}(\wp)\} \tag{2.1}$$

Δ_{ik} : The total cost of a shortest path from i to k.
c_{ij} : The cost from i to j.
P_{ik} : The set of all paths connecting i to k.
$\tilde{c}(\wp)$: The sum of all costs along path $\wp$.

Beginning the algorithm, the distance matrix $\Delta^{(0)}$ is initialised to the cost matrix C, as the shortest path between any pair of nodes without any intermediate node equals the initial cost between this pair (eq. 2.2).

$$\Delta_{ij}^{(0)} = c_{ij} \tag{2.2}$$

For the next step, it is considered to use 1 intermediate node. If a shortest path starting from node i, passing through any node j, and ends at k, Δ_{ik} equals Δ_{ij} plus Δ_{jk}. Taking this into account, equation 2.3 can be established.

$$\Delta_{ij}^{(1)} = min(\Delta_{ij}^{(0)}, \Delta_{i1}^{(0)} + \Delta_{1j}^{(0)}) \tag{2.3}$$

Continuing this process of thought, at iteration 2, 3, ..., t, the algorithm checks whether it is beneficial to include the additional intermediate node t in the already computed shortest paths between all pairs of nodes (eq. 2.4).

$$\Delta_{ij}^{(t)} = min(\Delta_{ij}^{t-1)}, \Delta_{i1}^{(t-1)} + \Delta_{1j}^{(t-1)}) \tag{2.4}$$

Assume that $\Delta_{ij}^{(t-1)}$ represents the shortest path between nodes i and j, considering only paths where an intermediate node is no higher than t-1, and therefore belongs to set {1, 2, ..., t-1}. When node t is added to the set, the new shortest path between i and j falls into one of the following scenarios:

- The path passes through t because it is shorter. In this case the shortest path distance becomes $\Delta_{ij}^{(t)} = \Delta_{it}^{(t)} + \Delta_{tj}^{(t)} = \Delta_{it}^{(t-1)} + \Delta_{tj}^{(t-1)}$.
- The path does not pass through t because doing so does not reduce the distance and thus the shortest path distance remains $\Delta_{it}^{(t-1)}$.

As a result, a sequence of matrices ($\Delta^{(0)}$, $\Delta^{(1)}$, ..., $\Delta^{(n)}$) is computed. The final matrix $\Delta^{(n)}$ contains the shortest path distances for all pairs, considering paths that can use all n nodes as intermediates. $\Delta^{(n)}$ represents the all-pairs shortest path distance matrix (eq. 2.5).

$$\Delta_{ij}^{(t)} = \begin{cases} c_{ij}, & \text{if } t = 0, \\ min(\Delta_{ij}^{(t-1)}, \Delta_{i1}^{(t-1)} + \Delta_{1j}^{(t-1)}) & \text{if } t \geq 1 \end{cases} \tag{2.5}$$

A basic implementation of the Floyd-Warschall algorithm is sufficient for the scope of this thesis. Nevertheless, it is important to note that this algorithm is part of the foundation for more advanced applications in graph theory and optimization. These advanced applications are introduced and explained in the work of F. Fouss, M. Saerens & M. Shimbo [11].

2.3.2 Randomised shortest path

In 2009, M. Saerens et al. published a work called: Randomised Shortest-Path Problems [18]. This work introduces a new approach to shortest-path algorithms, were a degree of randomness is added to the traditional approach. This concept, the randomised shortest-path problem, RSP, combines finding the lowest cost with a predetermined degree of exploration (entropy) within a network.

Its developed to handle situations where strictly determined paths could be a disadvantage. Traditional algorithms will look for the shortest path between two nodes, but are therefore stiff and predictable. A degree of randomness could be beneficial in a dynamic environment where links between nodes change. It could also introduce unpredictability in competitive scenarios or prevent congestion in logistical applications.

The work developed two models to handle the RSP:

- The fist model approaches the problem as a matter of optimisation. It will minimise the costs while maintaining a the expected entropy spread in the network.
- The second model, the Akamatsu model, uses the Boltzmann distribution to evaluate paths. The idea is that each path has a probability, based on the total costs of that path.

Looking at machine learning, the practical applications of graph theory are as prominent as ever. In this context, the algorithms proposed by Saerens offer intriguing possibilities. While comparing a ranking based on a RSP model with those from other mechanisms could provide interesting insights, it would likely be too abstract for basic management applications. As a results, the RSP theory is not explored deeper in this paper.

2.4 Elo rating

The Elo rating system, known to the general public mainly as a chess rating system, is a statistical method to calculate relative skill levels of players in a zero-sum game [10]. But besides chess, this system can be used to rank options when a pairwise comparison is possible. The system updates the rating of options based on the outcome of a pairwise comparison and the strength of the options.

An initial rating R_A and R_B is used to calculate an expected score E_A of option A against option B (eq. 2.6). Once the pairwise comparison is done, the actual result S_A is fixed. S_A equals 1 in case of a win, 0.5 for a draw and 0 in case of a loss. Subsequently, the ratings are updated, R', using R, S, E and an adjustment factor K (eq. 2.8).

Besides a ranking, bookmakers are fond of the Elo rating of teams in order help them to maintain a statistical advantage over their gamblers.

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} \tag{2.6}$$

$$E_B = E_A - 1 \tag{2.7}$$

$$R'_A = R_A + K(S_A - E_A) \tag{2.8}$$

$$R'_B = R_B + K(S_B - E_B) \tag{2.9}$$

2.4.1 Adjustment factor K

The adjustment factor K plays an important role on the sensitivity of the Elo ratings. A factor K that is too small will return ratings that update too slow and thus not adapt adequately to recent developments. On the other hand, if K is too large, recent results will have too much impact on the rating of a team.

The FIFA Women's World Ranking uses an Elo rating to form a ranking of women's national teams [12]. They use a different value for K, depending on the importance of the match. K varies from 15 for a friendly match to 60 for a World Cup match.

Another possibility is to search for the optimal parameters. The Elo rating system itself can be used to fine-tune the parameters of the system, using the mean squared error, MSE.

2.4.2 Actual result S

Based on solely the outcome, the most simple values for the actual results, S, are 1, 0.5 or 0 for a win, draw or loss. This can be ideal for a game with a ternary outcome like chess. However, when considering events where a margin of victory is involved, taking this margin into account can provide a more realistic representation of the match.

2.4.3 Mean Squared Error, MSE

The Mean Squared Error (eq. 2.10), MSE, is based on the delta between the expected result E and the actual result S. This criterion can be used to tune the parameters, like the adjustment factor K, of the Elo system. It can also serve as a metric for predictive capabilities of a system.

$$MSE = \frac{1}{n} \sum_i (E_i - S_i)^2 \tag{2.10}$$

A study that focusses purely on the implementation of the Elo rating in a football environment, can be found in the thesis of Thibault Gérard in 2019[13].

Comparison of ranking mechanisms

While most people don't realise it, rankings and ratings play a pivotal role in their everyday life. They enable meaningful comparisons and guide decisions across different domains: from politics, to business to consumption. This chapter studies different ranking mechanisms, using the Belgian Jupiler Pro League as a vehicle to examine their approach, mechanisms and implications.

A classic points-based system, where teams receive three, one or no points depending on the result of a match, has been around since 1995. It's a widely adopted and intuitive ranking mechanism, but in its simplicity it lacks some nuance to how a result is achieved and against which adversary. The Elo-rating system, commonly known from its use in chess, partially solves this lack for nuance. This dynamic system adapts the rating (and thus possibly the ranking) of each participant after a match based on the delta between expected outcome and actual outcome.

Looking into the domain of politics, the Schulze method is considered as a tool to rank football teams during a season. It considers each match as a pairwise comparison, interpreting the result as a preference for one team over the other. This method will find a strongest path based on graph theory, which could result in different conclusions compared to the classic points-based system.

Delving even deeper into graph theory, the Floyd-Warshall algorithm will search the most efficient path between two teams. To achieve this, the algorithm can exploit different available metrics of a match as weighted edges within a network. Its broad scope of application often supports backend calculations in diverse contexts. However, this chapter will evaluate whether the algorithm itself is suited for ranking football matches.

This chapter is inspired by "Who's #1?: The Science of Rating and Ranking" by Amy Langville [15]. Her work explores the principles behind ranking mechanisms in sports, academics and politics and emphasises the importance of understanding these principles to utilise these rankings. It provided inspiration to find an answer to the question: How do different pairwise comparison methods fare in a dynamic, competitive, real-world environment and what can we learn from their practical implementation?

3.1 Current points-based ranking system

The Belgian football competition, the Jupiler Pro League, uses a points based ranking system to determine the standings of its 18 teams throughout the season. Each team plays 34 matches, following a double round-robin format where every team competes twice against its competitors, once at home and once away. We do not consider the play-off system in this paper.

Teams earn points, based on the result of each match: three points for a win, one point for a draw, and no points for a loss. This structure, which rewards victories proportionally higher, incentivizes team to take additional risks to ensure a victory. All points accumulate to a score that is used to rank the teams and crown a national champion at the end of the season. Meanwhile, the lowest ranking team is relegated to the second tier of Belgian football.

It could occur that two or more teams have an equal amount of points, in which case their ranking is determined by additional metrics. The first tie-breaker is the goal difference, calculated as the delta between scored and conceded goals. In the odd chance teams remain tied, the total amount of goals scored is considered. These mitigations ensure that a final ranking is always established without ties.

Pos	Team	[B·O·W]	Wed	W	G	V	Ptn	DV	DT	+/−	
1	Club Brugge		34	24	4	6	**76**	73	26	+47	
2	Antwerp FC		34	18	6	10	**60**	57	48	+9	PO 1
3	RSC Anderlecht		34	15	13	6	**58**	51	34	+17	
4	KRC Genk		34	16	8	10	**56**	67	48	+19	
5	KV Oostende		34	15	8	11	**53**	49	41	+8	
6	Standard Luik		34	13	11	10	**50**	52	41	+11	
7	KAA Gent		34	14	7	13	**49**	55	42	+13	PO 2
8	KV Mechelen		34	13	9	12	**48**	54	54	0	
9	Beerschot VA		34	14	5	15	**47**	58	64	−6	
10	SV Zulte Waregem		34	14	4	16	**46**	53	69	−16	
11	OH Leuven		34	12	9	13	**45**	54	59	−5	
12	KAS Eupen		34	10	13	11	**43**	44	55	−11	
13	Sporting Charleroi		34	11	9	14	**42**	46	49	−3	
14	KV Kortrijk		34	11	6	17	**39**	44	57	−13	
15	STVV		34	10	8	16	**38**	41	52	−11	
16	Cercle Brugge		34	11	3	20	**36**	40	51	−11	
17	Waasland-Beveren		34	8	7	19	**31**	44	70	−26	
18	Excel Moeskroen		34	7	10	17	**31**	32	54	−22	

Table 3.1: JPL '20 - '21 official results [8]

Pos	Team	Wed	W	G	V	Ptn	DV	DT	+/−	
1	Union Sint-Gillis	34	24	5	5	**77**	78	27	+51	
2	Club Brugge	34	21	9	4	**72**	72	37	+35	PO 1
3	RSC Anderlecht	34	18	10	6	**64**	72	36	+36	
4	Antwerp FC	34	19	6	9	**63**	55	38	+17	
5	KAA Gent	34	18	8	8	**62**	56	30	+26	
6	Sporting Charleroi	34	15	9	10	**54**	55	46	+9	
7	KV Mechelen	34	15	7	12	**52**	57	61	−4	PO 2
8	KRC Genk	34	15	6	13	**51**	66	47	+19	
9	STVV	34	15	6	13	**51**	42	40	+2	
10	Cercle Brugge	34	12	9	13	**45**	49	46	+3	
11	OH Leuven	34	10	11	13	**41**	47	58	−11	
12	KV Oostende	34	10	7	17	**37**	34	61	−27	
13	KV Kortrijk	34	9	10	15	**37**	43	48	−5	
14	Standard Luik	34	9	9	16	**36**	32	51	−19	
15	KAS Eupen	34	8	8	18	**32**	37	61	−24	
16	SV Zulte Waregem	34	8	8	18	**32**	42	69	−27	
17	RFC Seraing	34	8	4	22	**28**	30	68	−38	
18	Beerschot VA	34	4	4	26	**16**	33	76	−43	

Table 3.2: JPL '21 - '22 official results [9]

This chapter will use the results of the 2020-21 (tab. 3.1) and 2021-22 (tab. 3.2) seasons to demonstrate and evaluate the different ranking mechanisms. The results of each match are visualised in matrix 3.3 and 3.4. A first interesting observation is that Union became champions in '22 over Bruges, despite them losing one game against Bruges and the other stranded in a draw.

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	W-B	ZWA
RSC Anderlecht		1–0	2–0	2–0	3–0	2–1	1–1	1–0	0–0	0–2	1–1	1–1	2–2	2–1	0–0	3–1	0–0	4–1
Antwerp FC	1–4		3–2	1–0	2–1	0–2	2–2	3–2	1–0	4–2	4–1	1–1	3–2	1–2	1–1	0–0	3–2	0–1
Beerschot VA	2–1	1–2		1–1	2–1	0–3	0–1	5–2	1–1	0–0	1–2	2–2	4–2	1–2	0–3	6–3	3–2	3–1
Cercle Brugge	0–0	2–1	2–1		3–4	1–2	1–2	1–5	5–2	0–1	0–1	1–2	3–0	0–1	0–1	3–0	2–0	1–3
Sporting Charleroi	1–0	2–0	3–1	3–0		1–1	2–3	1–2	0–1	0–0	0–1	1–1	1–1	1–0	1–2	0–0	0–2	1–1
Club Brugge	3–0	0–2	0–1	2–1	0–1		3–0	3–2	0–1	1–0	2–2	4–2	3–0	2–1	3–1	1–0	4–1	3–0
KAS Eupen	2–0	0–2	3–1	1–2	3–1	0–4		1–4	2–1	2–0	1–1	1–1	3–3	1–1	0–4	1–1	1–1	2–3
KRC Genk	1–2	4–2	1–2	2–0	2–1	1–2	4–0		1–1	2–0	3–1	4–1	1–1	2–2	2–2	4–0	1–1	3–2
KAA Gent	1–1	0–1	5–1	1–0	4–0	0–4	2–2	1–2		1–2	1–0	4–0	2–3	1–0	2–1	1–1	3–0	0–3
KV Kortrijk	1–3	1–3	5–5	1–2	1–3	1–2	0–0	2–1	1–0		1–4	3–0	0–3	3–1	2–1	0–2	1–3	1–2
KV Mechelen	2–2	3–0	2–3	2–3	3–3	0–3	3–0	0–0	1–1	1–2		2–1	2–2	0–1	0–4	2–0	2–3	4–2
Excel Moeskroen	1–1	2–3	3–1	1–2	1–1	0–0	0–2	2–0	0–1	0–3	0–1		2–2	0–1	1–0	3–2	1–1	4–2
OH Leuven	1–0	2–0	0–1	2–1	1–3	2–1	1–1	2–3	0–3	3–1	1–2	2–0		1–2	1–0	2–2	1–2	2–1
KV Oostende	2–2	1–1	1–2	1–1	3–2	1–3	1–1	3–1	2–1	2–1	2–0	3–0	3–1		2–2	3–1	0–2	3–0
Standard Luik	1–3	1–1	3–0	1–0	3–2	1–1	2–2	0–0	2–1	2–1	2–2	0–1	1–1	1–0		1–2	3–1	2–2
STVV	0–1	2–3	1–0	3–0	1–2	1–2	0–2	1–2	2–1	0–0	2–1	0–2	3–1	0–0	2–0		1–1	1–2
Waasland-Beveren	2–4	0–3	1–2	0–2	1–1	0–2	1–0	1–1	1–4	3–4	2–3	2–0	1–3	2–0	1–2	2–4		1–5
Zulte Waregem	2–2	1–3	0–3	1–0	0–2	0–6	2–1	1–2	2–7	1–1	1–2	1–0	2–3	2–1	3–2	0–2	4–1	

Table 3.3: JPL '20 - '21 Match matrix [8]

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
RSC Anderlecht		2–1	4–2	0–2	4–0	1–1	4–1	2–0	1–1[a]	1–1	7–2	2–2	3–0	3–0	1–1	2–0	1–3	3–2
Antwerp FC	2–0		2–1	1–1	3–0	1–1	4–2	4–2[b]	1–0	0–1	1–2	2–2	3–0	2–1	2–3	1–1	0–2	1–0
Beerschot VA	0–7	0–1		0–1[c]	2–3	1–3	0–3	2–0	0–2	2–1	0–1	3–1	0–2	3–0	0–1	0–1	0–3	3–3
Cercle Brugge	1–2	0–1	2–0		1–2	2–0	1–2	2–2	2–2	2–0	3–1	1–1	0–1	2–0	1–1	0–1	0–3	3–1
Sporting Charleroi	1–3	1–1	5–2	5–0		0–1	3–0	2–0	0–0	1–1	0–2	0–3	1–0	2–0	0–0	0–0	0–3	3–0
Club Brugge	2–2	4–1	3–2	1–1	2–0		2–2	3–1	1–2	2–0	2–0	1–1	3–0	3–2	2–2	2–0	0–0	3–0
KAS Eupen	1–1	0–1	1–0	0–2	0–4	1–3		3–2	0–1	2–2	1–1	3–1	0–2	1–2	0–2	2–1	2–3	1–1
KRC Genk	1–0	1–1	4–1	1–1	4–2	2–3	5–0		0–3	2–0	4–1	4–0	3–4	3–0	2–0	0–1	1–1	2–0
KAA Gent	1–0	0–1	2–2	2–1	2–3	6–1	2–0	1–0		2–2	2–0	5–0	1–1	4–0[d]	3–1	2–1	0–2	2–1
KV Kortrijk	2–3	0–2[e]	1–1	0–2	2–2	0–1	1–1	1–2	1–0		2–2	2–1	1–0	2–0	0–1	1–3	2–3	5–0
KV Mechelen	0–1	3–2	3–2	2–2	2–2	2–1	1–3	1–1	4–3	3–2		2–0	3–0	2–0	3–1	0–1	3–1	2–2
OH Leuven	0–0	0–1	0–0	3–2	1–1	1–4	1–4	2–1	0–1	2–1	5–0[f]		1–0	1–3	2–1	4–1	1–4	1–1
KV Oostende	2–2	1–2	3–1	2–1	0–3	1–3	2–1	0–4	1–0	0–2	2–4	1–3		2–2	1–0	0–0	1–7	0–2
RFC Seraing	0–5	0–1	2–1	2–1	1–3	0–5	0–0	0–2	0–0	0–2	1–0	1–1	2–3		0–1	2–0	0–4[g]	5–1
Standard Luik	0–1	2–5	1–0[h]	1–1	0–3[i]	2–2	1–0	1–1	0–1	1–1	1–2	2–2	1–0	0–1		1–2	1–3	0–1
STVV	2–2	2–1	3–2	1–2	0–1	1–2	2–0	1–2	2–1	0–2	1–1	2–0	1–1	3–1	3–0		1–2	1–3
Union Sint-Gillis	1–0	1–2	5–0[j]	3–2	4–0	0–1	0–0	2–1	0–0	2–0	2–0	1–3	1–1	4–2	4–0	0–1		2–1
Zulte Waregem	1–2	2–1	2–0	2–4	2–2	0–4	3–0	2–6	1–2	2–2	3–2	1–1	0–0[k]	1–0	1–2	0–2	0–2	

Table 3.4: JPL '21 - '22 Match matrix [9]

3.1.1 Cross validation Points-based ranking

In order to make statements about the precision of a ranking mechanism, it could be interesting to evaluate any predictive capabilities. To study these capabilities, cross validation can be introduced [6]. Cross validation is a statistical technique that parts the matches into a training set and a testing set. In this paper, a season is divided into 34 folds, one for each matchday. A sliding window loop is implemented to iteratively test the model's performance by training on 29 folds and testing on the remaining 5. After each iteration, the first fold of the current test set is moved to the training fold, while the next fold following the test set is added to the new test fold.

Note: Initially it was considered to divide the season in 5 training folds and 1 test fold, each consisting of 51 randomly selected matches. This would however not guarantee an equal amount of opportunities (matches) for every team. To work around this problem, a sliding window loop was programmed.

Two main evaluation metrics are used: Accuracy, which measures the percentage of correct predictions for win, draw or loss outcomes within the test set, and Mean Squared Error, MSE, which quantifies the difference between predicted score and actual score. These two metrics will be calculated for each ranking mechanism after a cross validation.

Accuracy

The accuracy is calculated by looking at the ability to predict match outcomes based on relative team points. A normalised points ratio (eq. 3.1) and a threshold for draws are incorporated in the calculations to better handle closely ranked teams. Its expected for the home team to win when the ratio exceeds 0.5 plus the threshold, t, and vice versa for the away team. If the ratio lies between $0.5 + t$ and $0.5 - t$, a draw is expected.

$$\text{normalised points ratio} = \frac{points_{home}}{points_{home} + points_{away}} \tag{3.1}$$

After running the Matlab script written for the cross validation (A.1), a mean accuracy of 41.63 % for the '20 - '21 season and 51.69 % for the '21 - '22 season can be found with a draw threshold of 0.025. This threshold results in a similar number of expected draws compared to the actual number of draws. One would initially think these are promising values, considering 3 possibilities (win, draw, loss) returns a baseline of 33.3 %. However, when comparing these accuracy values to a historical baseline of approximately 40 % victory for the home playing team, the conclusions for the two seasons are very different. While there is an outperformance of over 10 % for the '21 - '22 season, there is no outperformance for the season prior.

Mean Squared Error

The MSE was calculated (eq. 2.10) by looking at the difference between the normalised points ratio (eq. 3.1) and the actual scores, S, which were normalised through equation 3.2. After performing the cross validation, an MSE of 0.0293 for the '20 - '21 season and 0.0326 for the '21 - '22 season can be found.

$$S_{ij} = \frac{G_{ij} + 1}{G_{ij} + G_{ji} + 2} \tag{3.2}$$

G_{ij} : Number of goals scored by i against j.

The minor difference in MSE-values between the seasons suggests that the points based ranking has a robust predicting potential, regardless of variations in the matches. This consistency suggests that the difference between the predicted score and the actual score remains stable.

An average error of 0.173 ($= \sqrt{0.03}$) for each expected outcome that ranges between 0 and 1 is relatively low. To determine whether the system outperforms a baseline, the MSE is calculated for a constant expected score of 0.5. For the '20 - '21 season, the baseline returned a value of 0.0281, which is better than the MSE returned by the model. For the following season, the outperformance of our model compared to the baseline is negligible, 0.0326 to 0.0328.

Note that for the baseline with a constant expected score of 0.5, the accuracy drops severely as it would almost exclusively predict a draw result.

3.1.2 Conclusion

Predicting the result of a football match is no easy feat. If it were, bookmakers wouldn't be making the big profits they see today. They count on the unpredictability of sports, using this margin on top of their advanced prediction models to maintain a statistical advantage.

The difficulty is clearly visible in the previous section. Based on our cross validation of two seasons, the calculated expected scores based solely on the points based ranking are not entirely fit for the prediction of an actual outcome. Based on the two seasons, just a minor outperformance to the first baseline ($S_{home} = 1$) could be found for the accuracy and no outperformance to the second ($S_{home} = 0.5$) could be found for the MSE.

Note: The goal difference, used to calculate S, is often a measure of dominance during a match. However, this metric is also frequently influenced by external factors, causing an inevitable unknown. Finding a reduction in the MSE will thus be a difficult task, and will have its limitations, due the nature of a football season and a football match.

Nevertheless, the results of this section serve as a benchmark for comparisons with other ranking mechanisms that are discussed in later sections. By comparing these alongside the points based ranking, we have a standard to prevent ourselves from overcomplicating these more advanced mechanisms.

3.2 Elo rating

The classic points-based ranking is easy to grasp an universally known, but has some shortcomings. It will reward a victory against the lowest ranking team the same as it would a victory against the competition's leader. A chess rating, formally known as Elo-rating, can be introduced to address this problem. An Elo rating manages to take the relative strengths of two competing teams into account and indicates a form of momentum.

3.2.1 Fine-tuning the Elo rating

Model 1 - Basic model

Using the basic equations 2.6 and 2.8, a Matlab script can be written that goes over all 34 match-days of a season and calculates an Elo rating for each team (A.8). The adjustment factor K equals 30 and the partition of S equals [1, 0.5 and 0]. This results in the following ratings for 2020-2021 (tab. 3.5):

CLU	AND	ANT	OOS	GNT	KVM	GNK	STA	ZWA	EUP	OHL	BEE	STV	KVK	CER	CHA	MOE	WB
1648	1590	1574	1541	1538	1538	1529	1525	1497	1496	1468	1461	1461	1441	1439	1426	1418	1411

Table 3.5: JPL '20 - '21 Elo Ratings (Basic)

Model 2 - Fine-tuning margin of victory

Taking into account the margin of victory, a more realistic representation of the actual result, S, can be found (eq. 3.2). This equation considers the goals scored by i, G_{ij} and by j, G_{ij} and satisfies the criterion that $S_{ij} + S_{ji} = 1$. Keeping K equal to 30 results in the following ratings for 2020-2021 (tab. 3.6): [13]

CLU	AND	GNT	GNK	STA	ANT	KVM	OOS	EUP	OHL	ZWA	STV	CER	CHA	BEE	KVK	WB	MOE
1561	1528	1527	1521	1520	1511	1511	1510	1490	1488	1487	1486	1480	1480	1478	1478	1473	1469

Table 3.6: JPL '20 - '21 Elo Ratings (Including ΔG)

Model 3 - Fine-tuning adjustment factor K

The adjustment factor, which was initially set at 30, can be fine-tuned by using the MSE (eq. 2.10). A Matlab script (A.9) calculates the factor K which returns the lowest MSE-value. These calculations provide the following graph (tab. 3.1), which shows that $K = 19.1$ with a minimal mean square error equal to 0.0271. Assuming K equal to 19.1 results in the following ratings for 2020-2021 (tab. 3.7).

CLU	AND	GNT	GNK	STA	ANT	OOS	KVM	OHL	EUP	ZWA	STV	CHA	BEE	KVK	CER	WB	MOE
1550	1521	1519	1517	1514	1509	1508	1506	1493	1491	1490	1489	1489	1485	1484	1483	1478	1475

Table 3.7: JPL '20 - '21 Elo Ratings (Including ΔG & fine-tuned K)

Model 4 - Fine-tuning home advantage

In football, there is a certain home advantage is present. This advantage refers to the well documented trend that teams perform better when playing in their home stadium, in front of their own fans. There are several key factors. But regardless of the cause, a factor h can be added to the equation to account for this advantage (eq. 3.3).

$$E_A = \frac{1}{1 + 10^{(R_B - R_A - h)/400}} \tag{3.3}$$

A Matlab script (A.10) calculates the combination of K and h that return the lowest MSE. These calculations provide graph 3.2, which shows that for $K = 19.3$ and $h = 6.9$, the mean square error is minimal and equal to 0.0270. Performing these same computations for the 2021-2022 season, we find that $K = 49.3$, $h = 11.1$ with the minimal mean square error equal to 0.0.0278. Using these parameters returns the most refined Elo ratings (tab. 3.8/3.9).

Points based ranking	CLU	ANT	AND	GNK	OOS	STA	GNT	KVM	BEE	ZWA	OHL	EUP	CHA	KVK	STV	CER	WB	MOE
Points	76	60	58	56	53	50	49	48	47	46	45	43	42	39	38	36	31	31
Elo ranking	CLU	AND	GNT	GNK	STA	ANT	OOS	KVM	OHL	EUP	ZWA	STV	CHA	BEE	KVK	CER	WB	MOE
Elo rating	1.550	1.521	1.519	1.517	1.514	1.509	1.508	1.506	1.493	1.491	1.490	1.489	1.489	1.485	1.484	1.482	1.478	1.475

Table 3.8: JPL '20 - '21 Elo Ratings (Including ΔG, fine-tuned K & fine-tuned h)

Points based ranking	USG	CLU	AND	ANT	GNT	CHA	KVM	GNK	STV	CER	OHL	OOS	KVK	STA	EUP	ZWA	SER	BEE
Points	77	72	64	63	62	54	52	51	51	45	41	37	37	36	32	32	28	16
Elo ranking	USG	CLU	GNT	AND	ANT	GNK	STV	CER	CHA	KVM	KVK	OHL	STA	OOS	ZWA	EUP	BEE	SER
Elo rating	1.555	1.547	1.535	1.535	1.525	1.520	1.513	1.512	1.509	1.497	1.487	1.485	1.477	1.474	1.471	1.460	1.451	1.451

Table 3.9: JPL '21 - '22 Elo Ratings (Including ΔG, fine-tuned K & fine-tuned h)

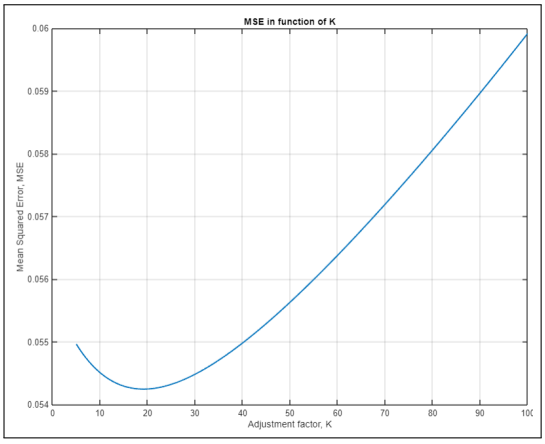

Figure 3.1: JPL '20 - '21 Optimisation K

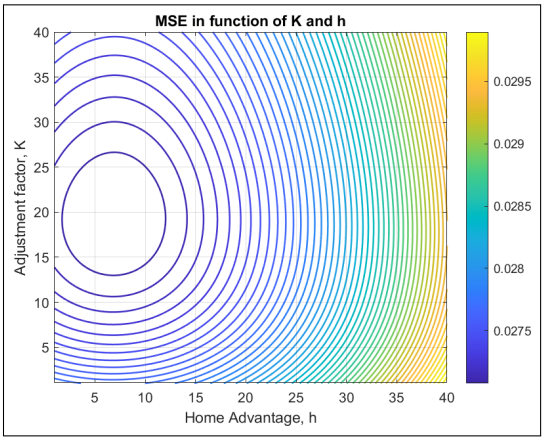

Figure 3.2: JPL '20 - '21 Optimisation K & h

The progression from the basic model to this more refined model, where parameters like a margin of victory are included and parameters like K and h are fine-tuned, should improve the precision as shown by the reduction of the MSE.

3.2.2 Cross validation Elo rating

Comparing the final Elo ranking with the original points based ranking, Club Brugges and Union still win the league (tab. 3.8/3.9). However, several teams see their ranking changing when switching mechanism.

In order to try and make a statement about which ranking is more correct, a cross validation is performed for the Elo rating. A season is once again divided into 34 folds, 29 of which are used as training sets and 5 consecutive folds as test set. The same two metrics are used as before: Accuracy, to measure the percentage of correct predictions, and MSE to quantify the difference between predicted score and actual score.

Accuracy

The accuracy was calculated by looking at the ability to predict match outcomes based on the Elo rating of the competing teams. All four models were used to display the effect of the score difference and the fine-tuning of the adjustment factor and the home advantage. Like before, its expected for the home team to win when the ratio exceeds 0.5 plus the threshold, t, and vice versa for the away team. If the ratio lies between 0.5 + t and 0.5 - t, a draw is expected.

	Elo Model 1	Elo Model 2	Elo Model 3	Elo Model 4
	- K = 30 - h = 0 - S = [1, 0.5, 0]	- K = 30 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 19,3 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 19,3 - h = 6,9 - S = (G_home + 1)/ (G_home+G_away+2)
	Accuracy	*Accuracy*	*Accuracy*	*Accuracy*
2020 - 2021	44,00%	43,04%	39,33%	38,37%
	- K = 30 - h = 0 - S = [1, 0.5, 0]	- K = 30 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 49,3 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 49,3 - h = 11,1 - S = (G_home + 1)/ (G_home+G_away+2)
	Accuracy	*Accuracy*	*Accuracy*	*Accuracy*
2021 - 2022	54,74%	49,67%	50,94%	53,18%

Table 3.10: Accuracy of Elo ranking after cross validation

A Matlab script (A.11) was written to perform the cross validation based on the Elo ratings. Table 3.10 shows very contrasting results:

- Despite the effort to progressively fine-tune the models, the basic model for '20 - '21 is notably better in the accurate prediction of a win, loss or draw. The reduction of the adjustment factor K, which was fine-tuned to find the minimum MSE, resulted in a lower accuracy. Implementing the home field advantage, h, further reduced the accuracy by 1%.
- Changing the interpretation of the actual result, S, from a win, draw or loss to a score difference to rate the training set, has a negative impact on the accuracy. This is most notable between model 1 and 2 for the '21 - '22 season. Fine-tuning K and h had a positive impact on the accuracy for that season, while model 1 remained the better option.
- We tweaked the Matlab code (A.11) to create a model where K and h are fine-tuned, but S remained 1, 0.5 or 0. For the '21 - '22 season, this resulted in an accuracy of 54.15%, which is 1% better than model 4. Still, it underperforms when looking at the basic model.

Mean Squared Error

The MSE is calculated by looking at the difference between the expected scores, E, (3.3) and the actual scores (3.2). Once again, all four models were used. Table 3.11 shows the results:

	Elo Model 1	Elo Model 2	Elo Model 3	Elo Model 4
	- K = 30 - h = 0 - S = [1, 0.5, 0]	- K = 30 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 19,3 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 19,3 - h = 6,9 - S = (G_home + 1)/ (G_home+G_away+2)
	MSE	*MSE*	*MSE*	*MSE*
2020 - 2021	0,0330	0,0267	0,0266	0,0264
	- K = 30 - h = 0 - S = [1, 0.5, 0]	- K = 30 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 49,3 - h = 0 - S = (G_home + 1)/ (G_home+G_away+2)	- K = 49,3 - h = 11,1 - S = (G_home + 1)/ (G_home+G_away+2)
	MSE	*MSE*	*MSE*	*MSE*
2021 - 2022	0,0357	0,0270	0,0277	0,0274

Table 3.11: MSE of Elo ranking after cross validation

– Very similar values can be found for the MSE between model 2, 3 and 4, regardless of the season. This is due to the interpretation of the actual result S. Using the goal difference instead of just the outcome [1, 0.5, 0] notably improves the MSE.

– A slight improvement over model 2 and 3 can be seen in the '20 - '21 season when using the custom parameters. These values for K and h were calculated with the results of all the matches (tab. 3.1), which means they are not guaranteed to provide the minimum MSE for the test set. However, given that the training set covers most of the matches (29 out of 34), these parameters will return a value close to the minimum MSE.

– The fact that the parameters are not fine-tuned specifically for the test set, is the reason we see a near stagnant MSE between model 2, 3 and 4 for the '21 - '22 season.

Elo rating versus points based ranking

	Baseline (assuming S_home = 1)		Baseline (assuming S_home = 0,5)		Points based ranking		Elo Model 4		Elo Model 1	
	Accuracy	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*
2020-2021	40,74%	0,2659	23,48%	0,0281	41,63%	0,0293	38,37%	0,0264	44,00%	0,0330
2021-2022	42,40%	0,2683	21,85%	0,0328	51,69%	0,0316	53,18%	0,0274	54,74%	0,0357

Table 3.12: Elo rating versus points based ranking after cross validation

Both Elo rating models demonstrate their potential compared to the points based ranking and the baseline (tab. 3.12). The basic Elo model notably triumphs both the Points based ranking and the baseline that assumes home victory in accuracy. The fine-tuned model on the other hand excels in minimising the Mean Squared Error. It manages to beat the baseline, which the Points based ranking could not achieve.

3.2.3 Conclusion Elo rating

Although the traditional points based ranking is more intuitive and easier to grasp for the general public, the Elo ranking is likely to be a better representation of the relative strength between teams. Further research across more seasons can further solidify the conviction that the Elo rating outperforms the points based ranking in terms of MSE.

Both systems have difficulties with the accurate prediction of an outcome, especially in the '20 - '21 season. This demonstrates the complexity of a football match on the one hand, but it also shows that both systems, with the limited parameters now taken into account, are inadequate in accurately predicting an actual outcome.

3.3 Schulze method

It was established earlier in section 2.1.4 that the Schulze method is a ranked voting system that aims to find a ranking based on pairwise comparisons between all options. The basic idea behind the Schulze method is that we use the pairwise preferences of the voters to construct a graph. Then the preferences between candidates are found by tracing paths on that graph. It's often referred to as the "Beatpath method".

Each option will compete head-to-head with all its alternatives, creating a beatpath if it can indirectly beat another option through a chain of wins. The relative strength of a path relies on the weakest of its links. The Schulze method will ensure a final ranking that respects the transitive strength of each option over its alternatives. If we were to replace the preference after a voting round with a goal difference after a match, we could use the system to determine a champion after a football season.

3.3.1 Establish a ranking based on the Schulze method

Initialise the "Pairwise preference matrix"

In reality, we do not have a preference between teams but rather a score difference. With 18 teams playing each other twice, once at home and once away, this results in 34 rounds and thus 306 unique matches in a season (tab. 3.1/3.2). These outcomes are presented in a pairwise preference matrix (tab. 3.13/3.14), P, where P_{ij} represents the score difference between the home team and away team. A negative value indicates a win for the away team, while a draw is represented by a zero score.

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND		1	2	2	3	1	0	1	0	-2	0	0	0	1	0	2	0	3
ANT	-3		1	1	1	-2	0	1	1	2	3	0	1	-1	0	0	1	-1
BEE	1	-1		0	1	-3	-1	3	0	0	-1	0	2	-1	-3	3	1	2
CER	0	1	1		-1	-1	-1	-4	3	-1	-1	-1	3	-1	-1	3	2	-2
CHA	1	2	2	3		0	-1	-1	-1	0	-1	0	0	1	-1	0	-2	0
CLU	3	-2	-1	1	-1		3	1	-1	1	0	2	3	1	2	1	3	3
EUP	2	-2	2	-1	2	-4		-3	1	2	0	0	0	0	-4	0	0	-1
GNK	-1	2	-1	2	1	-1	4		0	2	2	3	0	0	0	4	0	1
GNT	0	-1	4	1	4	-4	0	-1		-1	1	4	-1	1	1	0	3	-3
KVK	-2	-2	0	-1	-2	-1	0	1	1		-3	3	-3	2	1	-2	-2	-1
KVM	0	3	-1	-1	0	-3	3	0	0	-1		1	0	-1	-4	2	-1	2
MOE	0	-1	2	-1	0	0	-2	2	-1	-3	-1		0	-1	1	1	0	2
OHL	1	2	-1	1	-2	1	0	-1	-3	2	-1	2		-1	1	0	-1	1
OOS	0	0	-1	0	1	-2	0	2	1	1	2	3	2		0	2	-2	3
STA	-2	0	3	1	1	0	0	0	1	1	0	-1	0	1		-1	2	0
STV	-1	-1	1	3	-1	-1	-2	-1	1	0	1	-2	2	0	2		0	-1
WB	-2	-3	-1	-2	0	-2	1	0	-3	-1	-1	2	-2	2	-1	-2		-4
ZWA	0	-2	-3	1	-2	-6	1	-1	-5	0	-1	1	-1	1	1	-2	3	

Table 3.13: JPL '20 - '21
Pairwise preference matrix

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND		1	2	-2	4	0	3	2	0	0	5	0	3	3	0	2	-2	1
ANT	2		1	0	3	0	2	2	1	-1	-1	0	3	1	-1	0	-2	1
BEE	-7	-1		-1	-1	-2	-3	2	-2	1	-1	2	-2	3	-1	-1	-3	0
CER	-1	-1	2		-1	2	-1	0	0	2	2	0	-1	2	0	-1	-3	2
CHA	-2	0	3	5		-1	3	2	0	0	-2	-3	1	2	0	0	-3	3
CLU	0	3	1	0	2		0	2	1	2	2	0	3	1	0	2	0	3
EU	0	-1	1	-2	-4	-2		1	-1	0	0	2	-2	-1	-2	1	-1	0
GNK	1	0	3	0	-2	-1	5		-3	2	3	4	-1	3	2	-1	0	2
GNT	1	-1	0	1	-1	5	2	1		0	2	5	0	4	2	1	-2	1
KVK	-1	-2	0	-2	0	-1	0	-1	1		0	1	1	2	-1	-2	-1	5
KVM	-1	-1	1	0	0	1	-2	0	1	1		2	3	2	2	-1	2	0
OHL	0	-1	0	1	0	-3	-3	1	-1	1	5		1	-2	1	3	-3	0
OOS	0	-1	2	1	-3	-2	1	-4	1	-2	-2	-2		0	1	0	-6	-2
SER	-5	-1	1	1	-2	-5	0	-2	0	-2	1	0	-1		-1	2	-4	4
STA	-1	-3	1	0	-3	0	1	0	-1	0	-2	0	1	-1		-1	-2	-1
STV	0	1	1	-1	-1	-1	2	-1	1	-2	0	2	0	2	3		-1	-2
USG	1	-1	5	1	4	-1	0	1	0	2	2	-2	0	2	4	-1		1
ZWA	-1	1	2	-2	0	-4	3	-4	-1	0	1	0	0	1	-1	-2	-2	

Table 3.14: JPL '21 - '22
Pairwise preference matrix

Establish the "Preference graph"

This pairwise preference matrix can be transformed to a graph where teams are represented through the nodes, with the weight of the links representing the margins of preferences, in our case the goal difference over the two matches. Two transposed cells, representing the home and away match between two teams, are subtracted from each other. If the result is negative, it is replace by the value -∞, which implies there is no edge between the nodes.

A preference graph with 18 nodes (fig. B.1/B.2) is difficult to depict on an A4 format in a functional manner (A.4). Therefore, table 3.15/3.16 displays the graph in a clean and workable format.

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	-Inf	4	1	2	2	-Inf	-Inf	2	0	0	0	0	-Inf	1	2	3	2	3
ANT	-Inf	-Inf	2	0	-Inf	0	2	-Inf	2	4	0	1	-Inf	-Inf	0	1	4	1
BEE	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	4	-Inf	0	0	-Inf	3	0	-Inf	2	2	5
CER	-Inf	0	1	-Inf	-Inf	-Inf	0	-Inf	2	0	0	0	2	-Inf	-Inf	0	4	-Inf
CHA	-Inf	1	1	4	-Inf	1	-Inf	-Inf	-Inf	2	-Inf	0	2	0	-Inf	1	-Inf	2
CLU	2	0	2	2	-Inf	-Inf	7	2	3	2	3	2	2	3	2	2	5	9
EUP	2	-Inf	3	0	3	-Inf	-Inf	-Inf	1	2	-Inf	2	0	0	-Inf	2	-Inf	-Inf
GNK	-Inf	1	-Inf	6	2	-Inf	7	-Inf	1	1	2	1	1	-Inf	0	5	0	2
GNT	0	-Inf	4	-Inf	5	-Inf	-Inf	-Inf	-Inf	-Inf	1	5	2	0	0	-Inf	6	2
KVK	0	-Inf	0	0	-Inf	-Inf	-Inf	-Inf	2	-Inf	-Inf	6	-Inf	1	0	-Inf	-Inf	-Inf
KVM	0	0	0	0	1	-Inf	3	-Inf	-Inf	2	-Inf	2	1	-Inf	-Inf	1	0	3
MOE	0	-Inf	2	0	0	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	2	3	-Inf	1
OHL	1	1	-Inf	-Inf	-Inf	-Inf	0	-Inf	-Inf	5	-Inf	2	-Inf	-Inf	1	-Inf	1	2
OOS	-Inf	1	0	1	0	-Inf	0	2	0	-Inf	3	4	3	-Inf	-Inf	2	-Inf	2
STA	-Inf	0	6	2	2	-Inf	4	0	0	0	4	-Inf	-Inf	1	-Inf	-Inf	3	-Inf
STV	-Inf	-Inf	-Inf	0	-Inf	-Inf	-Inf	-Inf	1	2	-Inf	-Inf	2	-Inf	3	-Inf	2	1
WB	-Inf	-Inf	-Inf	-Inf	2	-Inf	1	0	-Inf	1	0	2	-Inf	4	-Inf	-Inf	-Inf	-Inf
ZWA	-Inf	-Inf	-Inf	3	-Inf	-Inf	2	-Inf	-Inf	1	-Inf	-Inf	-Inf	-Inf	1	-Inf	7	-Inf

Table 3.15: JPL '20 - '21
Preference graph visualised as matrix

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	NaN	-Inf	9	-Inf	6	0	3	1	-Inf	1	6	0	3	8	1	2	-Inf	2
ANT	1	NaN	2	1	3	-Inf	3	2	2	1	0	1	4	2	2	-Inf	-Inf	0
BEE	-Inf	-Inf	NaN	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	1	-Inf	2	-Inf	2	-Inf	-Inf	-Inf	-Inf
CER	1	-Inf	3	NaN	-Inf	2	1	0	-Inf	4	2	-Inf	-Inf	1	0	0	-Inf	4
CHA	-Inf	-Inf	4	6	NaN	-Inf	7	4	1	0	-Inf	-Inf	4	4	3	1	-Inf	3
CLU	0	3	3	-Inf	3	NaN	2	3	-Inf	3	1	3	5	6	0	3	1	7
EU	-Inf	-Inf	4	-Inf	-Inf	-Inf	NaN	-Inf	-Inf	0	2	5	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf
GNK	-Inf	-Inf	1	0	-Inf	-Inf	4	NaN	-Inf	3	3	3	3	5	2	0	-Inf	6
GNT	1	-Inf	2	1	-Inf	4	3	4	NaN	-Inf	1	6	-Inf	4	3	0	-Inf	2
KVK	-Inf	-Inf	-Inf	-Inf	0	-Inf	0	-Inf	1	NaN	-Inf	0	3	4	-Inf	0	-Inf	5
KVM	-Inf	0	2	-Inf	2	-Inf	-Inf	-Inf	-Inf	1	NaN	-Inf	5	1	4	-Inf	0	-Inf
OHL	0	-Inf	-Inf	1	3	-Inf	-Inf	-Inf	-Inf	0	3	NaN	3	-Inf	1	1	-Inf	0
OOS	-Inf	-Inf	4	2	-Inf	-Inf	3	-Inf	1	-Inf	-Inf	NaN	1	0	0	-Inf	-Inf	-Inf
SER	-Inf	-Inf	-Inf	-Inf	-Inf	-Inf	1	-Inf	-Inf	-Inf	-Inf	2	-Inf	NaN	0	0	-Inf	3
STA	-Inf	-Inf	2	0	-Inf	0	3	-Inf	-Inf	1	-Inf	-Inf	0	0	NaN	-Inf	-Inf	0
STV	-Inf	1	2	0	-Inf	-Inf	1	0	0	0	1	-Inf	0	0	4	NaN	0	0
USG	3	1	8	4	7	-Inf	1	1	2	3	0	1	6	6	6	0	NaN	3
ZWA	-Inf	0	2	-Inf	-Inf	-Inf	3	-Inf	-Inf	-Inf	1	0	2	-Inf	0	0	-Inf	NaN

Table 3.16: JPL '21 - '22
Preference graph visualised as matrix

Calculate strongest paths (A.3)

The strength of a path is defined by the weakest links that make up the path. It indicates the robustness of a preference across all possible paths, ensuring the winner is the option that would most strongly defeat all others in a direct confrontation whilst taking indirect confrontations into account.

Initially, the strength between two nodes is based on their direct connection, if one exists. We then iteratively calculate the strongest paths by introducing more intermediate nodes. With each additional node, we examine all pairs and update their strongest link. Once all nodes are checked, the strongest paths between every pair of nodes is established (tab. 3.17/3.18). Teams with consistently higher values in their row will generally be ranked higher.

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	-Inf	4	3	3	3	1	3	3	2	4	3	4	3	4	3	3	4	3
ANT	2	-Inf	3	3	3	1	3	3	2	4	3	4	3	4	3	3	4	3
BEE	2	2	-Inf	4	3	1	4	4	2	3	3	4	3	4	3	4	5	5
CER	2	2	3	-Inf	3	1	3	3	2	3	3	4	3	4	3	3	4	3
CHA	2	2	3	4	-Inf	1	3	3	2	3	3	4	3	4	3	3	4	3
CLU	2	2	3	3	3	-Inf	7	3	3	3	3	4	3	4	3	3	7	9
EUP	2	2	3	3	3	1	-Inf	3	2	3	3	3	3	3	3	3	3	3
GNK	2	2	3	6	3	1	7	-Inf	2	3	3	4	3	4	3	5	4	3
GNT	2	2	4	4	5	1	4	4	-Inf	3	3	5	3	4	3	4	6	4
KVK	2	2	3	3	3	1	3	3	2	-Inf	3	6	3	3	3	3	3	3
KVM	2	2	3	3	3	1	3	3	2	3	-Inf	3	3	3	3	3	3	3
MOE	2	2	3	3	3	1	3	3	2	3	3	-Inf	3	3	3	3	3	3
OHL	2	2	3	3	3	1	3	3	2	5	3	5	-Inf	3	3	3	3	3
OOS	2	2	3	3	3	1	3	3	2	3	3	4	3	-Inf	3	3	3	3
STA	2	2	6	4	3	1	4	4	2	3	4	4	3	4	-Inf	4	5	5
STV	2	2	3	3	3	1	3	3	2	3	3	4	3	3	3	-Inf	3	3
WB	2	2	3	3	3	1	3	3	2	3	3	4	3	4	3	3	-Inf	3
ZWA	2	2	3	3	3	1	3	3	2	3	3	4	3	4	3	3	7	-Inf

Table 3.17: JPL '20 - '21 Strongest path matrix

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	-Inf	2	9	6	6	2	6	4	2	4	6	5	5	8	4	2	1	4
ANT	1	-Inf	4	3	3	2	3	3	2	3	3	3	4	3	3	2	1	3
BEE	1	2	-Inf	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2
CER	1	2	3	-Inf	3	2	3	3	2	4	3	3	3	4	3	2	1	4
CHA	1	2	4	6	-Inf	2	7	4	2	4	3	5	4	4	3	2	1	4
CLU	1	3	4	3	3	-Inf	3	3	2	3	3	3	5	6	3	3	1	7
EUP	1	2	4	3	3	2	-Inf	3	2	3	3	5	3	3	3	2	1	3
GNK	1	2	4	3	3	2	4	-Inf	2	3	3	4	3	5	3	2	1	6
GNT	1	3	4	3	3	4	4	4	-Inf	3	3	6	4	4	3	3	1	4
KVK	1	2	3	3	3	2	3	3	2	-Inf	3	3	3	4	3	2	1	5
KVM	1	2	4	3	3	2	3	3	2	3	-Inf	3	5	3	4	2	1	3
OHL	1	2	3	3	3	2	3	3	2	3	3	-Inf	3	3	3	2	1	3
OOS	1	2	4	3	3	2	3	3	2	3	3	3	-Inf	3	3	2	1	3
SER	1	2	3	3	3	2	3	3	2	3	3	3	3	-Inf	3	2	1	3
STA	1	2	3	3	3	2	3	3	2	3	3	3	3	3	-Inf	2	1	3
STV	1	2	3	3	3	2	3	3	2	3	3	3	3	3	4	-Inf	1	3
USG	3	2	8	6	7	2	7	4	2	4	3	5	6	6	6	2	-Inf	4
ZWA	1	2	3	3	3	2	3	3	2	3	3	3	3	3	3	2	1	-Inf

Table 3.18: JPL '21 - '22 Strongest path matrix

Establish a final ranking

A final ranking is established by counting how many matches a team performed better than every other team (tab. 3.19/3.20). The team which beats the highest number of other teams is ranked first. For 2020-2021, it is established that Club Brugges wins, with Anderlecht comming second whilst Antwerp and Ghent share third place. For 2021-2022, the competition would be won by Union, with Anderlecht and Gent following second and third.

Points based ranking	CLU	ANT	AND	GNK	OOS	STA	GNT	KVM	BEE	ZWA	OHL	EUP	CHA	KVK	STV	CER	WB	MOE
Points	76	60	58	56	53	50	49	48	47	46	45	43	42	39	38	36	31	31
Elo ranking	CLU	AND	GNT	GNK	STA	ANT	OOS	KVM	OHL	EUP	ZWA	STV	CHA	BEE	KVK	CER	WB	MOE
Elo rating	1.550	1.521	1.519	1.517	1.514	1.509	1.508	1.506	1.493	1.491	1.490	1.489	1.489	1.485	1.484	1.482	1.478	1.475
Schulze ranking	CLU	AND	GNT	ANT	STA	BEE	GNK	CHA	ZWA	CER	OHL	WB	KVK	OOS	EUP	KVM	MOE	STV
Dominance score	17	15	14	14	10	8	6	4	3	3	2	2	1	1	0	0	0	0
Total path strength	65	53	61	51	60	56	58	50	52	49	50	48	49	47	46	46	46	46

Table 3.19: JPL '20 - '21 Ranking of options according to the Schulze method

Points based ranking	USG	CLU	AND	ANT	GNT	CHA	KVM	GNK	STV	CER	OHL	OOS	KVK	STA	EUP	ZWA	SER	BEE
Points	77	72	64	63	62	54	52	51	51	45	41	37	37	36	32	32	28	16
Elo ranking	USG	CLU	GNT	AND	ANT	GNK	STV	CER	CHA	KVM	KVK	OHL	STA	OOS	ZWA	EUP	BEE	SER
Elo rating	1.555	1.547	1.535	1.535	1.525	1.520	1.513	1.512	1.509	1.497	1.487	1.485	1.477	1.474	1.471	1.460	1.451	1.451
Schulze ranking	USG	AND	GNT	CLU	ANT	STV	CHA	GNK	CER	KVM	KVK	EUP	OOS	OHL	SER	STA	ZWA	BEE
Dominance score	17	16	15	14	12	12	9	5	4	3	3	2	1	1	1	1	1	0
Total path strength	77	76	57	56	46	45	58	51	46	47	46	46	44	43	43	43	43	32

Table 3.20: JPL '21 - '22 Ranking of options according to the Schulze method

Looking at the '20-'21 season, Club Brugges is ranked the strongest team, as it dominates all 17 other teams. This suggests Club has a stronger direct or indirect path over every other team. It reflects a consistent performance either directly by winning games, or indirectly by beating teams that perform well against others.

Antwerp and Gent have an equal dominance score, which results in a tie. In an election format, this would be rare due to the essence of the pairwise strongest paths. However, a football competition will only directly compare two teams twice in 34 rounds. This, combined with the small goal differences, results in close values in the pairwise preference matrix, which increases the chances for ties.

There is no imposed method to breaking ties. In this example however, the total path strength could be a secondary metric to rate and rank the teams. It is the sum of the strongest paths from a team to every other team. It can be found for any team by accumulating all values of its row from the strongest path matrix (tab. 3.17/3.18).

3.3.2 Cross validation Schulze ranking

If we want to use the cross validation on the Schulze ranking like previously for the points based ranking and the Elo ranking, we will encounter an obstacle: One of the strengths of the Schulze method, is the non-uniform contribution of matches. A match between two title contenders contributes significantly to the strongest path matrix. When all the data is available or a large amount of pairwise comparisons are made between two options, the Schulze method can produce a solid ranking.

In one football season however, only two matches (pairwise comparisons) are played between every possible pairing. Therefore, it's inevitable that some matches that would have a significant impact on the strongest path matrix, are not included in the training set. This could lead to a distorted ranking being used for the training set.

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	0	1	2	2	3	0	0	1	0	-2	0	0	0	1	0	2	0	0
ANT	0	0	1	1	1	-2	0	0	1	2	3	0	1	-1	0	0	1	-1
BEE	1	-1	0	0	0	-3	-1	3	0	0	-1	0	2	0	-3	3	1	2
CER	0	1	0	0	-1	-1	0	-4	3	-1	-1	-1	0	-1	-1	3	2	-2
CHA	1	2	2	3	0	0	0	-1	-1	0	-1	0	0	1	-1	0	-2	0
CLU	3	0	-1	1	-1	0	3	1	-1	1	0	0	3	1	2	1	3	3
EUP	2	-2	2	-1	2	-4	0	-3	1	0	0	0	0	0	0	0	0	-1
GNK	-1	2	-1	2	1	-1	4	0	0	2	2	3	0	0	0	0	0	1
GNT	0	-1	4	0	0	-4	0	-1	0	-1	1	4	-1	1	1	0	3	-3
KVK	-2	-2	0	-1	-2	0	0	0	1	0	0	3	-3	2	1	-2	-2	-1
KVM	0	0	-1	-1	0	-3	3	0	0	-1	0	1	0	-1	-4	2	-1	0
MOE	0	0	2	-1	0	0	-2	2	-1	-3	-1	0	0	0	1	1	0	0
OHL	0	2	-1	1	-2	1	0	0	-3	2	-1	2	0	-1	1	0	0	1
OOS	0	0	-1	0	0	-2	0	2	1	1	2	3	2	0	0	2	0	3
STA	-2	0	0	1	1	0	0	0	0	1	0	-1	0	1	0	-1	0	0
STV	0	-1	1	3	-1	0	-2	-1	1	0	0	-2	2	0	2	0	0	-1
WB	-2	-3	-1	-2	0	-2	1	0	-3	0	-1	2	-2	2	-1	0	0	-4
ZWA	0	-2	-3	1	-2	-6	0	-1	0	0	-1	0	-1	1	1	-2	3	0

Table 3.21: JPL '20 - '21 Pairwise preference matrix of a training set

Table 3.21 shows the pairwise preference matrix of a random training set. All matches that belong to the subsequent test set are marked yellow. These marked matches are thus seen as "no preference" for the training set. This set results in a recognisable but altered strongest path matrix (tab. 3.22). Table 3.23 indicates that some crucial matches for Anderlecht must have been missing in the training set, as it goes from second in the original ranking to last in the training ranking.

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	- Inf	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	2	2
ANT	2	- Inf	3	3	2	1	3	3	3	4	3	4	3	2	3	3	4	3
BEE	2	2	- Inf	4	2	1	4	4	3	3	3	3	3	2	3	3	5	5
CER	2	2	3	- Inf	2	1	3	3	3	3	3	3	3	2	3	3	4	3
CHA	2	2	3	4	- Inf	1	3	3	3	3	3	3	3	2	3	3	4	3
CLU	3	2	3	3	2	- Inf	7	3	3	3	3	3	3	3	3	3	7	9
EUP	2	2	3	3	2	1	- Inf	3	3	3	3	3	3	2	3	3	3	3
GNK	2	2	3	6	2	1	7	- Inf	3	3	3	3	3	2	3	3	4	3
GNT	2	2	4	4	2	1	4	4	- Inf	3	3	5	3	2	3	3	6	4
KVK	2	2	3	3	2	1	3	3	3	- Inf	3	6	3	2	3	3	3	3
KVM	2	2	3	3	2	1	3	3	3	3	- Inf	3	3	2	3	3	3	3
MOE	2	2	3	3	2	1	3	3	3	3	3	- Inf	3	2	3	3	3	3
OHL	2	2	3	3	2	1	3	3	3	5	3	5	- Inf	2	3	3	3	3
OOS	2	2	3	3	2	1	3	3	3	3	3	3	3	- Inf	3	3	3	3
STA	2	2	3	3	2	1	3	3	3	3	4	3	3	2	- Inf	3	3	3
STV	2	2	3	3	2	1	3	3	3	3	3	3	3	2	3	- Inf	3	3
WB	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	2	- Inf	2
ZWA	2	2	3	3	2	1	3	3	3	3	3	3	3	2	3	3	7	- Inf

Table 3.22: JPL '20 - '21 Strongest path matrix random training set

Complete Schulze ranking	CLU	AND	GNT	ANT	STA	BEE	GNK	CHA	ZWA	CER	OHL	WB	KVK	OOS	EUP	KVM	MOE	STV
Dominance score	17	15	14	14	10	8	6	4	3	3	2	2	1	1	0	0	0	0
Total path strength	65	53	61	51	60	56	58	50	52	49	50	48	49	47	46	46	46	46
Random training set Schulze ranking	CLU	ANT	CHA	OOS	GNT	BEE	GNK	OHL	KVK	STA	ZWA	CER	EUP	KVM	MOE	STV	AND	WB
Dominance score	17	13	13	13	7	5	3	3	2	2	1	1	1	1	1	1	0	0
Total path strength	63	49	48	46	55	53	53	49	48	46	49	46	45	45	45	45	33	33

Table 3.23: JPL '20 - '21 Ranking according to Schulze method - Entire set vs random training set

This example shows the effect on a single fold. But for a football season with limited pairwise comparisons, it's inevitable that this occurs in other folds as well.

Accuracy

The accuracy was determined by looking at the ability to predict match outcomes based on the Schulze ranking. The strongest path matrix returns a dominance score and total path strength, based on which a expected result can be calculated. The team with the higher dominance score is expected to win. If this score is a tie, the ratio of both team's total path strengths are used. the home team is expected to win when the ratio exceeds 0.5 plus threshold, t, and vice versa for the away team. If the ratio lies between 0.5 + t and 0.5 - t, a draw is expected.

	Baseline (assuming S_home = 1)		Baseline (assuming S_home = 0,5)		Points based ranking		Elo Model 4		Elo Model 1		Schulze method	
	Accuracy	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*
2020-2021	40,74%	0,2659	23,48%	0,0281	41,63%	0,0293	38,37%	0,0264	44,00%	0,0330	42,07%	0,0834
2021-2022	42,40%	0,2683	21,85%	0,0328	51,69%	0,0316	53,18%	0,0274	54,74%	0,0357	50,34%	0,0616

Table 3.24: Schulze method versus Elo rating versus points based ranking after cross validation

Despite the more complex calculations (A.5), table 3.24 indicates that the Schulze method produces similar results to those of the points based ranking. This lack in outperformance is most likely due to the training sets missing matches in each fold (tab. 3.23).

The potential of the Schulze method becomes more evident when the cross validation loop is allowed to use the entire dataset for training. By using the dominance scores and total path strengths after 34 folds, the accuracy improves to 48.07 % for '20 - '21 and to 55.78 % for '21 - '22 (tab. 3.25). This was done without altering the computation method.

	Schulze method (29 folds as training set)		Schulze method (34 folds as training set)	
	Accuracy	*MSE*	*Accuracy*	*MSE*
2020-2021	42,07%	0,0834	48,07%	0,0834
2021-2022	50,34%	0,0616	55,78%	0,0574

Table 3.25: Schulze method: partial training set vs complete training set after cross validation

While this better accuracy is achieved with the benefit of hindsight, it shows the potential of the Schulze method. When applied in a format with a large number of pairwise comparisons between two options, the distortion in the training set ranking is minimised and a decent accuracy can be expected.

Mean Squared Error

Table 3.24 shows an underperformance of the Schulze method in minimising the MSE. The main cause lies in the calculation of the expected outcome E. While the actual outcome S is derived from the goals scored by both teams (eq. 3.2), E in the Schulze method is based on the Dominance score (eq. 3.4). This difference leads to greater variation in E compared to S, resulting in a larger MSE.

$$E_{home} = \frac{DominanceScore_{home} + 1}{DominanceScore_{home} + DominanceScore_{away} + 2} \tag{3.4}$$

To illustrate this, consider the '20-'21 season match where Club Brugges defeated Moeskroen with 4-2.

- Club Brugges won the Points based ranking with 76 points.
- Club Brugges won the Schulze ranking with a dominance score of 17.
- Moeskroen was shared last in the Points based ranking with 31 points.
- Moeskroen was shared last in the Schulze ranking with a dominance score of 0.

Both mechanisms predict a win for Brugges, but the magnitude of E differs significantly: $E_{Schulze} = 0.95$ - $E_{Points-based} = 0.71$. The actual match result 4-2 gives $S = 0.625$, which is a lot closer to the points based prediction. This example illustrates the tendency of the Schulze method to expect more extreme results (closer to 1 or 0) due to the disparities in the Dominance scores. In contrast, the points based ranking and the Elo ranking have a tendency to expect more moderate results (closer to 0.5). Since S tends to cluster around moderate values, the Schulze method's polarised expected outcomes contribute to a larger MSE.

3.3.3 Conclusion Schulze method

The Schulze method is not very transparent compared to the previous mechanisms. It is however very good at addressing more complex interdependencies by considering both direct and indirect results. When Schulze is applied to a football season, it successfully identifies a representative ranking.

However, as we have limited ourselves to a single football season at a time, the mechanism has to rely on just two pairwise comparisons between each pairing of teams. Key matches

missing from the training set distort training rankings, reducing prediction accuracy during the cross validation. Additionally, the computation of the expected outcome contributes to a higher Mean Squared Error compared to simpler methods like the points based ranking or the Elo ranking.

Despite these limitations, the potential of the Schulze method becomes evident when applied to a more complete dataset, yielding a better accuracy than the other mechanisms for both seasons. It's reasonable to expect this trend to continue to scenarios involving a lot more pairwise comparisons between each pairing, without relying on the benefit of hindsight. In these cases, the Schulze method could provide a reliable framework to determine expected outcomes.

3.4 Floyd-Warshall algorithm

The Floyd-Warshall algorithm provides the computational backbone for the Schulze method. But while the Schulze method is looking for the strongest paths between two teams, the Floyd-Warshall algorithm will be solely interested in the shortest paths. Its scope of application is very broad, commonly used in scenarios that require efficient pathfinding. We will examine if the algorithm has any potential as a ranking mechanism.

3.4.1 Establish a ranking based on the Floyd-Warshall algorithm

Initialise the Pairwise preference matrix

A pairwise preference matrix, which comes down to a score difference matrix in this environment, is established to represent the outcome of the 306 unique matches over a season (tab. 3.13/3.14).

Initialise the Cost matrix

Cost matrix C is an interpretation of the Pairwise preference matrix. Based on the outcome or score of a match, a cost c_{ij} is determined for going from node i to node j. For the most basic approach, only the outcome will be considered:

- Team i won against team j: $c_{ij} = 1$
- Team i played a draw against team j: $c_{ij} = 3$
- Team i lost against team j: $c_{ij} = 5$

Home / Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	Inf	1	1	1	1	1	3	1	3	5	3	3	3	1	3	1	3	1
ANT	5	Inf	1	1	1	5	3	1	1	1	1	3	1	5	3	3	1	5
BEE	1	5	Inf	3	1	5	5	1	3	3	5	3	1	5	5	1	1	1
CER	3	1	1	Inf	5	5	5	5	1	5	5	5	1	5	5	1	1	5
CHA	1	1	1	1	Inf	3	5	5	5	3	5	3	3	1	5	3	5	3
CLU	1	5	5	1	5	Inf	1	1	5	1	3	1	1	1	1	1	1	1
EUP	1	5	1	5	1	5	Inf	5	1	1	3	3	3	3	5	3	3	5
GNK	5	1	5	1	1	5	1	Inf	3	1	1	1	3	3	3	1	3	1
GNT	3	5	1	1	1	5	3	5	Inf	5	1	1	5	1	1	3	1	5
KVK	5	5	3	5	5	5	3	1	1	Inf	5	1	5	1	1	5	5	5
KVM	3	1	5	5	3	5	1	3	3	5	Inf	1	3	5	5	1	5	1
MOE	3	5	1	5	3	3	5	1	5	5	5	Inf	3	5	1	1	3	1
OHL	1	1	5	1	5	1	3	5	5	1	5	1	Inf	5	1	3	5	1
OOS	3	3	5	3	1	5	3	1	1	1	1	1	1	Inf	3	1	5	1
STA	5	3	1	1	1	3	3	3	1	1	3	5	3	1	Inf	5	1	3
STV	5	5	1	1	5	5	5	5	1	3	1	5	1	3	1	Inf	3	5
WB	5	5	5	5	3	5	1	3	5	5	5	1	5	1	5	5	Inf	5
ZWA	3	5	5	1	5	5	1	5	5	3	5	1	5	1	1	5	1	Inf

Table 3.26: JPL '20 - '21 Cost matrix

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	Inf	1	1	5	1	3	1	1	3	3	1	3	1	1	3	1	5	1
ANT	1	Inf	1	3	1	3	1	1	1	5	5	3	1	1	5	3	5	1
BEE	5	5	Inf	5	5	5	5	1	5	1	5	1	5	1	5	5	5	3
CER	5	5	1	Inf	5	1	5	3	3	1	1	3	5	1	3	5	5	1
CHA	5	3	1	1	Inf	5	1	1	3	3	5	5	1	1	3	3	5	1
CLU	3	1	1	3	1	Inf	3	1	5	1	1	3	1	1	3	1	3	1
EUP	3	5	1	5	5	5	Inf	1	5	3	3	1	5	5	5	1	5	3
GNK	1	3	1	3	1	5	1	Inf	5	1	1	5	1	1	5	3	1	1
GNT	1	5	3	1	5	1	1	1	Inf	3	1	1	3	1	1	1	5	1
KVK	5	5	3	5	3	5	3	5	1	Inf	3	1	1	1	5	5	5	1
KVM	5	1	1	3	3	1	5	3	1	1	Inf	1	1	1	1	5	1	3
OHL	3	5	3	1	3	5	5	1	5	1	1	Inf	1	5	1	1	5	3
OOS	3	5	1	1	5	5	1	5	1	5	5	5	Inf	3	1	3	5	5
SER	5	5	1	1	5	5	3	5	3	5	1	3	5	Inf	5	1	5	1
STA	5	5	1	3	5	3	1	3	5	3	5	3	1	5	Inf	5	5	5
STV	3	1	1	5	5	5	1	5	1	5	3	1	3	1	1	Inf	5	5
USG	1	5	1	1	1	5	3	1	3	1	1	5	3	1	1	5	Inf	1
ZWA	5	1	1	5	3	5	1	5	5	3	1	3	3	1	5	5	5	Inf

Table 3.27: JPL '21 - '22 Cost matrix

Establish the "Distance matrix"

With this cost matrix C (tab. 3.26/3.27), the Floyd-Warshall algorithm can be applied in search for the shortest paths. Distance matrix $\Delta^{(0)}$ is initialised to C as it's the shortest path

between teams without an intermediate node. Further iterations progressively consider the use of additional intermediate nodes. After n iterations, n equalling the amount of teams, a final Distance matrix $\Delta^{(n)}$ (tab. 3.28/3.29) contains the shortest paths for all pairs (eq. 2.4).

Home / Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	0	1	1	1	1	1	2	1	2	2	2	2	2	1	2	1	2	1
ANT	2	0	1	1	1	2	2	1	1	1	1	2	1	2	2	2	1	2
BEE	1	2	0	2	1	2	2	1	2	2	2	2	1	2	2	1	1	1
CER	2	1	1	0	2	2	2	2	1	2	2	2	1	2	2	1	1	2
CHA	1	1	1	1	0	2	3	2	2	2	2	2	2	1	3	2	2	2
CLU	1	2	2	1	2	0	1	1	2	1	2	1	1	1	1	1	1	1
EUP	1	2	1	2	1	2	0	2	1	1	2	2	2	2	2	2	2	2
GNK	2	1	2	1	1	3	1	0	2	1	1	1	2	2	2	1	2	1
GNT	2	2	1	1	1	3	2	2	0	2	1	1	2	1	1	2	1	2
KVK	3	2	2	2	2	3	2	1	1	0	2	1	2	1	1	2	2	2
KVM	2	1	2	2	2	3	1	2	2	2	0	1	2	2	2	1	2	1
MOE	2	2	1	2	2	3	2	1	2	2	2	0	2	2	1	1	2	1
OHL	1	1	2	1	2	1	2	2	2	1	2	1	0	2	1	2	2	1
OOS	2	2	2	2	1	2	2	1	1	1	1	1	1	0	2	1	2	1
STA	2	2	1	1	1	3	2	2	1	1	2	2	2	1	0	2	1	2
STV	2	2	1	1	2	2	2	2	1	2	1	2	1	2	1	0	2	2
WB	2	3	2	3	2	3	1	2	2	2	2	1	2	1	2	2	0	2
ZWA	2	2	2	1	2	3	1	2	2	2	2	1	2	1	1	2	1	0

Table 3.28: JPL '20 - '21 Distance matrix

Thuis \ Uit	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	0	1	1	2	1	2	1	1	2	2	1	2	1	1	2	1	2	1
ANT	1	0	1	2	1	2	1	1	1	2	2	2	1	1	2	2	3	1
BEE	2	3	0	2	2	3	2	1	2	1	2	1	2	1	2	2	3	2
CER	3	2	1	0	2	1	2	2	2	1	1	2	2	1	2	2	2	1
CHA	2	2	1	1	0	2	1	1	2	2	2	2	2	1	1	2	3	1
CLU	2	1	1	2	1	0	2	1	2	1	2	1	1	1	2	1	2	1
EUP	2	2	1	2	2	3	0	1	2	2	2	1	2	2	2	1	3	2
GNK	1	2	1	2	1	2	1	0	2	1	1	1	2	1	1	2	2	1
GNT	1	2	2	1	2	1	1	1	0	2	1	1	2	1	1	1	2	1
KVK	2	2	2	2	3	2	2	2	1	0	2	1	1	1	2	2	3	1
KVM	2	1	1	2	2	1	2	2	1	1	0	1	1	1	1	2	1	2
OHL	2	2	2	1	2	2	2	1	2	1	1	0	1	2	1	1	2	2
OOS	2	3	1	1	3	2	1	2	1	2	2	2	0	2	1	2	3	2
SER	3	2	1	1	3	2	2	2	2	1	2	2	2	0	2	1	2	1
STA	3	3	1	2	3	3	1	2	2	2	3	2	1	2	0	2	4	3
STV	2	1	1	2	2	2	1	2	1	2	2	1	2	1	1	0	3	2
USG	1	2	1	1	2	2	1	2	1	2	1	2	1	1	2	2	0	1
ZWA	2	1	1	2	2	2	1	2	2	2	1	2	2	1	2	2	2	0

Table 3.29: JPL '21 - '22 Distance matrix

Establish a ranking

There are several approaches to establishing a Floyd-Warshall ranking based on $\Delta^{(n)}$. In this example, it's opted to use the aggregate distances, which comes down to the sum of all shortest paths from a team to every other team. A low aggregate distance indicates a consistently shorter path and thus a more direct dominance to other teams. A team with a high aggregate distance fails to achieve dominance and is further from its competitors. The aggregate distance for each team can be interpreted as the row-wise sum of $\Delta^{(n)}$.

Due to the nature of a football season, the spread of the aggregate distances is limited, which naturally results in ties. The inbound distance, which is the sum of all shortest paths from every other team to a given team, can serve as a tie-breaker. A high inbound distance for a team suggests many other teams have a longer path towards this team, reflecting a competitive ranking. The inbound distance for each team can be interpreted as the column-wise sum of $\Delta^{(n)}$.

Looking at the established rankings (tab. 3.30/3.31), a change in the season's champion can be seen for the '21-'22 season. The top six is closely matched, with Ghent narrowly outperforming Union with one point. Note that, if Ghent had scored one point less, it would have dropped to fourth as the three following teams have a greater inbound distance.

Points based ranking	CLU	ANT	AND	GNK	OOS	STA	GNT	KVM	BEE	ZWA	OHL	EUP	CHA	KVK	STV	CER	WB	MOE
Points	76	60	58	56	53	50	49	48	47	46	45	43	42	39	38	36	31	31
Elo ranking	CLU	AND	GNT	GNK	STA	ANT	OOS	KVM	OHL	EUP	ZWA	STV	CHA	BEE	KVK	CER	WB	MOE
Elo rating	1.550	1.521	1.519	1.517	1.514	1.509	1.508	1.506	1.493	1.491	1.490	1.489	1.489	1.485	1.484	1.482	1.478	1.475
Schulze ranking	CLU	AND	GNT	ANT	STA	BEE	GNK	CHA	ZWA	CER	OHL	WB	KVK	OOS	EUP	KVM	MOE	STV
Dominance score	17	15	14	14	10	8	6	4	3	3	2	2	1	1	0	0	0	0
Total path strength	65	53	61	51	60	56	58	50	52	49	50	48	49	47	46	46	46	46
Floyd-Warshall ranking	CLU	AND	ANT	OOS	OHL	GNK	GNT	BEE	STA	STV	CER	EUP	ZWA	KVM	MOE	KVK	CHA	WB
Aggregate distance	22	25	25	25	26	26	27	27	28	28	28	29	29	30	30	31	31	34
Inbound distance	40	30	29	26	28	27	27	25	28	26	25	30	26	29	25	27	26	27

Table 3.30: JPL '20 - '21 Ranking of options according to the Floyd-Warshall algorithm

Points based ranking	USG	CLU	AND	ANT	GNT	CHA	KVM	GNK	STV	CER	OHL	OOS	KVK	STA	EUP	ZWA	SER	BEE
Points	77	72	64	63	62	54	52	51	51	45	41	37	37	36	32	32	28	16
Elo ranking	USG	CLU	GNT	AND	ANT	GNK	STV	CER	CHA	KVM	KVK	OHL	STA	OOS	ZWA	EUP	BEE	SER
Elo rating	1.555	1.547	1.535	1.535	1.525	1.520	1.513	1.512	1.509	1.497	1.487	1.485	1.477	1.474	1.471	1.460	1.451	1.451
Schulze ranking	USG	AND	GNT	CLU	ANT	STV	CHA	GNK	CER	KVM	KVK	EUP	OOS	OHL	SER	STA	ZWA	BEE
Dominance score	17	16	15	14	12	12	9	5	4	3	3	2	1	1	1	1	1	0
Total path strength	77	76	57	56	46	45	58	51	46	47	46	46	44	43	43	43	43	32
Floyd-Warshall ranking	GNT	USG	CLU	AND	KVM	GNK	ANT	OHL	CHA	STV	CER	ZWA	KVK	SER	OOS	EUP	BEE	STA
Aggregate distance	23	24	24	24	24	24	26	27	28	28	29	29	31	31	32	32	33	39
Inbound distance	29	42	34	33	26	25	32	27	33	28	28	25	27	21	26	25	20	27

Table 3.31: JPL '21 - '22 Ranking of options according to the Floyd-Warshall algorithm

3.4.2 Establish a ranking based on the Floyd-Warshall algorithm using a goal based cost matrix

Initialise the "Pairwise preference matrix"

See table (3.13/3.14).

Initialise the "Cost matrix"

In contrast to before, not just the outcome but the actual goal difference will be taken into account. In case of a victory or loss by one goal, the same cost as before will be allocated to a win (eq. 3.5) or a loss (eq. 3.7). Also for a draw (eq. 3.6), the cost remains three. A larger goal difference however results in a lower cost for a win or a higher cost for a loss.

$$c_{ij} = \frac{2}{\text{Goal difference} + 1} \qquad \text{In case of a home win} \tag{3.5}$$

$$c_{ij} = 3 \qquad \text{In case of a draw} \tag{3.6}$$

$$c_{ij} = 2.5 \cdot (|\text{Goal difference}| + 1) \qquad \text{In case of a home loss} \tag{3.7}$$

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	Inf	1,00	0,67	0,67	0,50	1,00	3,00	1,00	3,00	7,50	3,00	3,00	3,00	1,00	3,00	0,67	3,00	0,50
ANT	10,00	Inf	1,00	1,00	1,00	7,50	3,00	1,00	1,00	0,67	0,50	3,00	1,00	5,00	3,00	3,00	1,00	5,00
BEE	1,00	5,00	Inf	3,00	1,00	10,00	5,00	0,50	3,00	3,00	5,00	3,00	0,67	5,00	10,00	0,50	1,00	0,67
CER	3,00	1,00	1,00	Inf	5,00	5,00	5,00	12,50	0,50	5,00	5,00	5,00	0,50	5,00	5,00	0,50	0,67	7,50
CHA	1,00	0,67	0,67	0,50	Inf	3,00	5,00	5,00	5,00	3,00	5,00	3,00	3,00	1,00	5,00	3,00	7,50	3,00
CLU	0,50	7,50	5,00	1,00	5,00	Inf	0,50	1,00	5,00	1,00	3,00	0,67	0,50	1,00	0,67	1,00	0,50	0,50
EUP	0,67	7,50	0,67	5,00	0,67	12,50	Inf	10,00	1,00	0,67	3,00	3,00	3,00	12,50	3,00	3,00	5,00	
GNK	5,00	0,67	5,00	0,67	1,00	5,00	0,40	Inf	3,00	0,67	0,67	0,50	3,00	3,00	3,00	0,40	3,00	1,00
GNT	3,00	5,00	0,40	1,00	0,40	12,50	3,00	5,00	Inf	5,00	1,00	0,40	5,00	1,00	1,00	3,00	0,50	10,00
KVK	7,50	7,50	3,00	5,00	7,50	5,00	3,00	1,00	1,00	Inf	10,00	0,50	10,00	0,67	1,00	7,50	7,50	5,00
KVM	3,00	0,50	5,00	5,00	3,00	10,00	0,50	3,00	3,00	5,00	Inf	1,00	3,00	5,00	12,50	0,67	5,00	0,67
MOE	3,00	5,00	0,67	5,00	3,00	3,00	7,50	0,67	5,00	10,00	5,00	Inf	3,00	5,00	1,00	1,00	3,00	0,67
OHL	1,00	0,67	5,00	1,00	7,50	1,00	3,00	5,00	10,00	0,67	5,00	0,67	Inf	5,00	1,00	3,00	5,00	1,00
OOS	3,00	3,00	5,00	3,00	1,00	7,50	3,00	0,67	1,00	1,00	0,67	0,50	0,67	Inf	3,00	0,67	7,50	0,50
STA	7,50	3,00	0,50	1,00	1,00	3,00	3,00	3,00	1,00	1,00	3,00	5,00	3,00	1,00	Inf	5,00	0,67	3,00
STV	5,00	5,00	1,00	0,50	5,00	5,00	7,50	5,00	1,00	3,00	1,00	7,50	0,67	3,00	0,67	Inf	3,00	5,00
WB	7,50	10,00	5,00	7,50	3,00	7,50	1,00	3,00	10,00	5,00	5,00	0,67	7,50	0,67	5,00	7,50	Inf	12,50
ZWA	3,00	7,50	10,00	1,00	7,50	17,50	1,00	5,00	15,00	3,00	5,00	1,00	5,00	1,00	1,00	7,50	0,50	Inf

Table 3.32: JPL '20 - '21
Goal based Cost matrix

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	Inf	1,00	0,67	7,50	0,40	3,00	0,50	0,67	3,00	3,00	0,33	3,00	0,50	0,50	3,00	0,67	7,50	1,00
ANT	0,67	Inf	1,00	3,00	0,50	3,00	0,67	0,67	1,00	5,00	5,00	3,00	0,50	1,00	5,00	3,00	7,50	1,00
BEE	20,00	5,00	Inf	5,00	5,00	7,50	10,00	0,67	7,50	1,00	5,00	0,67	7,50	0,50	5,00	5,00	10,00	3,00
CER	5,00	5,00	0,67	Inf	5,00	0,67	5,00	3,00	3,00	0,67	0,67	3,00	5,00	0,67	3,00	5,00	10,00	0,67
CHA	7,50	3,00	0,50	0,33	Inf	5,00	0,50	0,67	3,00	3,00	7,50	10,00	1,00	0,67	3,00	3,00	10,00	0,50
CLU	3,00	0,50	1,00	3,00	0,67	Inf	3,00	0,67	5,00	0,67	0,67	3,00	0,50	1,00	3,00	0,67	3,00	0,50
EUP	3,00	5,00	1,00	7,50	12,50	7,50	Inf	1,00	5,00	3,00	3,00	0,67	7,50	5,00	7,50	1,00	5,00	3,00
GNK	1,00	3,00	0,50	3,00	0,67	5,00	0,33	Inf	10,00	0,67	0,50	0,40	5,00	0,50	0,67	5,00	3,00	0,67
GNT	1,00	5,00	3,00	1,00	5,00	0,33	0,67	1,00	Inf	3,00	0,67	0,33	3,00	0,40	0,67	1,00	7,50	1,00
KVK	5,00	7,50	3,00	7,50	3,00	5,00	3,00	5,00	1,00	Inf	3,00	1,00	1,00	0,67	5,00	7,50	5,00	0,33
KVM	5,00	1,00	1,00	3,00	3,00	1,00	7,50	3,00	1,00	1,00	Inf	0,67	0,50	0,67	0,67	5,00	0,67	3,00
OHL	3,00	5,00	3,00	1,00	3,00	10,00	10,00	1,00	5,00	1,00	0,33	Inf	1,00	7,50	1,00	0,50	10,00	3,00
OOS	3,00	5,00	0,67	1,00	10,00	7,50	1,00	12,50	1,00	7,50	7,50	7,50	Inf	3,00	1,00	3,00	17,50	7,50
SER	15,00	5,00	1,00	1,00	7,50	15,00	3,00	7,50	3,00	7,50	1,00	3,00	5,00	Inf	5,00	0,67	12,50	0,40
STA	5,00	10,00	1,00	3,00	10,00	3,00	1,00	3,00	5,00	3,00	5,00	3,00	1,00	5,00	Inf	5,00	7,50	5,00
STV	3,00	1,00	1,00	5,00	5,00	5,00	0,67	5,00	1,00	7,50	3,00	0,67	3,00	0,67	0,50	Inf	5,00	7,50
USG	1,00	5,00	0,33	1,00	0,40	5,00	3,00	1,00	3,00	0,67	0,67	7,50	3,00	0,67	0,40	5,00	Inf	1,00
ZWA	5,00	1,00	0,67	7,50	3,00	12,50	0,50	12,50	5,00	3,00	1,00	3,00	3,00	1,00	5,00	7,50	7,50	Inf

Table 3.33: JPL '21 - '22
Goal based Cost matrix

Establish the "Distance matrix"

With the Cost matrix based on the goal difference (tab. 3.32/3.33), the Floyd-Warshall algorithm can be applied once again in search for the shortest paths (tab. 3.34/3.35).

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	0	1,00	0,67	0,67	0,50	1,00	1,40	1,00	1,17	1,67	1,50	1,50	1,17	1,00	1,33	0,67	1,00	0,50
ANT	1,67	0	1,00	1,00	1,00	2,00	1,00	1,00	1,00	0,67	0,50	1,17	1,00	1,33	1,67	1,17	1,00	1,17
BEE	1,00	1,17	0	1,00	1,00	1,67	0,90	0,50	1,50	1,17	1,17	1,00	0,67	1,67	1,17	0,50	1,00	0,67
CER	1,50	1,00	0,90	0	0,90	1,50	1,67	1,40	0,50	1,17	1,50	0,90	0,50	1,33	1,17	0,50	0,67	1,50
CHA	1,00	0,67	0,67	0,50	0	2,00	1,57	1,17	1,00	1,33	1,17	1,40	1,00	1,00	1,67	1,00	1,17	1,33
CLU	0,50	1,17	1,17	1,00	1,00	0	0,50	1,00	1,50	1,00	1,67	0,67	0,50	1,00	0,67	1,00	0,50	0,50
EUP	0,67	1,33	0,67	1,17	0,67	1,67	0	1,17	1,00	0,67	1,83	1,17	1,33	1,33	1,67	1,17	1,50	1,17
GNK	1,07	0,67	1,07	0,67	1,00	2,07	0,40	0	1,17	0,67	0,67	0,50	1,07	1,33	1,07	0,40	1,33	1,00
GNT	1,40	1,07	0,40	0,90	0,40	2,07	1,30	0,90	0	1,57	1,00	0,40	1,07	1,00	1,00	0,90	0,50	1,07
KVK	2,07	1,67	1,17	1,67	1,40	2,33	1,40	1,00	1,00	0	1,33	0,50	1,33	0,67	1,00	1,33	1,50	1,17
KVM	1,17	0,50	1,17	1,17	1,17	2,17	0,50	1,50	1,50	1,17	0	1,00	1,33	1,67	1,33	0,67	1,17	0,67
MOE	1,67	1,33	0,67	1,33	1,67	2,33	1,07	0,67	1,83	1,33	1,33	0	1,33	1,67	1,00	1,00	1,17	0,67
OHL	1,00	0,67	1,33	1,00	1,50	1,00	1,50	1,33	1,50	0,67	1,17	0,67	0	1,33	1,00	1,50	1,50	1,00
OOS	1,67	1,17	1,17	1,17	1,00	1,67	1,07	0,67	1,00	1,00	0,67	0,50	0,67	0	1,33	0,67	1,00	0,50
STA	1,50	1,67	0,50	1,00	1,00	2,17	1,40	1,00	1,00	1,00	1,67	1,33	1,17	1,00	0	1,00	0,67	1,17
STV	1,67	1,33	1,00	0,50	1,40	1,67	1,50	1,50	1,00	1,33	1,00	1,33	0,67	1,67	0,67	0	1,17	1,67
WB	1,67	1,83	1,33	1,83	1,67	2,33	1,00	1,33	1,67	1,67	1,33	0,67	1,33	0,67	1,67	1,33	0	1,17
ZWA	1,67	2,00	1,50	1,00	1,67	2,50	1,00	1,67	1,50	1,67	1,67	1,00	1,50	1,00	1,00	1,50	0,50	0

Table 3.34: JPL '20 - '21
Goal based Distance matrix

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	OHL	OOS	SER	STA	STV	USG	ZWA
AND	0	1,00	0,67	0,73	0,40	1,33	0,50	0,67	1,33	1,33	0,33	1,00	0,50	0,50	1,00	0,67	1,00	0,90
ANT	0,67	0	1,00	0,83	0,50	1,33	0,67	0,67	1,00	1,33	1,00	1,07	0,50	1,00	1,33	1,33	1,67	1,00
BEE	1,67	1,90	0	1,50	1,33	2,00	1,00	0,67	2,00	1,00	1,00	0,67	1,50	0,50	1,33	1,17	1,67	0,90
CER	1,83	1,17	0,67	0	1,33	0,67	1,17	1,33	1,67	0,67	0,67	1,33	1,17	0,67	1,33	1,33	1,33	0,67
CHA	1,67	1,50	0,50	0,33	0	1,00	0,50	0,67	2,00	1,00	1,00	1,07	1,00	0,67	1,33	1,33	1,67	0,50
CLU	1,17	0,50	1,00	1,00	0,67	0	1,00	0,67	1,50	0,67	0,67	1,07	0,50	1,00	1,17	0,67	1,33	0,50
EUP	2,00	2,00	1,00	1,67	1,67	2,00	0	1,00	2,00	1,67	1,00	0,67	1,50	1,50	1,50	1,00	1,67	1,67
GNK	1,00	1,50	0,50	1,00	0,67	1,50	0,33	0	1,50	0,67	0,50	0,40	1,00	0,50	0,67	0,90	1,17	0,67
GNT	1,00	0,83	1,33	1,00	1,00	0,33	0,67	1,00	0	1,00	0,67	0,33	0,83	0,40	0,67	0,83	1,33	0,80
KVK	2,00	1,33	1,00	1,67	1,83	1,33	0,83	1,67	1,00	0	1,33	1,00	1,00	0,67	1,67	1,33	2,00	0,33
KVM	1,67	1,00	1,00	1,40	1,07	1,00	1,50	1,67	1,00	1,00	0	0,67	0,50	0,67	0,67	1,17	0,67	1,07
OHL	2,00	1,33	1,33	1,00	1,40	1,33	1,17	1,00	1,33	1,00	0,33	0	0,83	1,00	1,00	0,50	1,00	1,33
OOS	2,00	1,83	0,67	1,00	2,00	1,33	1,00	1,33	1,00	1,67	1,67	1,33	0	1,17	1,00	1,83	2,33	1,57
SER	2,07	1,40	1,00	1,00	1,90	1,67	0,90	1,67	1,67	1,67	1,00	1,33	1,50	0	1,17	0,67	1,67	0,40
STA	2,67	2,83	1,00	2,00	2,33	2,33	1,00	1,67	2,00	2,00	2,00	1,67	1,00	1,50	0	2,00	2,67	1,90
STV	1,67	1,00	1,00	1,67	1,50	1,33	0,67	1,67	1,00	1,67	1,00	0,67	1,50	0,67	0,50	0	1,67	1,07
USG	1,00	1,67	0,33	0,73	0,40	1,40	0,90	1,00	1,67	0,67	0,67	1,00	1,17	0,67	0,40	1,33	0	0,90
ZWA	1,67	1,00	0,67	1,83	1,50	2,00	0,50	1,33	2,00	1,67	1,00	1,17	1,50	1,00	1,67	1,50	1,67	0

Table 3.35: JPL '21 - '22
Goal based Distance matrix

Establish a ranking

Once again, the aggregate distances are calculated to establish a Floyd-Warshall ranking. A low aggregate distance could indicate that on average a team managed to win matches more dominantly.

Comparing both Floyd-Warshall rankings over a season (tab. 3.36/3.37), hardly any team maintains its ranking when changing to the goal based cost. The teams are clustered together, but due to the calculation of the cost, ties are much less likely. Regardless, the inbound distance is still calculated to break-up any ties. To make a statement about wheter or not these rankings mechanisms are representative, a cross validation will be performed.

Floyd-Warshall ranking *Outcome based cost [1, 3, 5]*	CLU	AND	ANT	OOS	OHL	GNK	GNT	BEE	STA	STV	CER	EUP	ZWA	KVM	MOE	KVK	CHA	WB
Aggregate distance	22	25	25	25	26	26	27	27	28	28	28	29	29	30	30	31	31	34
Inbound distance	40	30	29	26	28	27	27	25	28	26	25	30	26	29	25	27	26	27
Floyd-Warshall ranking *Goal based cost*	CLU	GNK	OOS	GNT	AND	BEE	CER	ANT	CHA	OHL	KVM	EUP	STA	STV	MOE	KVK	ZWA	WB
Aggregate distance	15,33	16,13	16,90	16,93	17,73	17,73	18,60	19,33	19,63	19,67	19,83	20,17	20,23	21,07	22,07	22,53	24,33	24,50
Inbound distance	32,13	18,80	20,67	20,83	22,87	16,37	17,57	20,23	18,93	17,63	21,17	19,17	20,40	16,30	15,70	19,73	16,90	17,33

Table 3.36: JPL '20 - '21 Floyd-Warshall ranking based on outcome and on goal difference

Floyd-Warshall ranking *Outcome based cost [1, 3, 5]*	GNT	USG	CLU	AND	KVM	GNK	ANT	OHL	CHA	STV	CER	ZWA	KVK	SER	OOS	EUP	BEE	STA
Aggregate distance	23	24	24	24	24	24	26	27	28	28	29	29	31	31	32	32	33	39
Inbound distance	29	42	34	33	26	25	32	27	33	28	28	25	27	21	26	25	20	27
Floyd-Warshall ranking *Goal based cost*	AND	GNT	GNK	CLU	USG	ANT	KVM	CHA	OHL	CER	STV	BEE	KVK	SER	ZWA	OOS	EUP	STA
Aggregate distance	13,87	14,03	14,47	15,07	15,90	16,90	17,70	17,73	18,90	19,00	20,23	21,80	22,00	22,67	23,67	24,73	25,50	32,57
Inbound distance	27,73	25,67	19,67	23,90	26,50	23,80	15,83	21,50	16,43	20,37	19,57	14,67	20,67	14,07	16,17	17,50	14,30	18,40

Table 3.37: JPL '21 - '22 Floyd-Warshall ranking based on outcome and on goal difference

3.4.3 Cross validation Floyd-Warshall ranking (A.7)

The Floyd-Warshall ranking, like the Schulze ranking, relies on direct and indirect paths. Therefore they could encounter similar difficulties when it comes to a cross validation. In a situation like a football season, where data is sparse, the training set will inevitably miss important results. These missing results will be interpreted in the cost matrix as an infinite cost. Considering the same training set as for the previous illustration (tab. 3.21), the following cost matrix (tab. 3.38) and subsequently the distance matrix (tab. 3.39) can be found:

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	0	1	1	1	1	Inf	3	1	3	5	3	3	3	1	3	1	3	Inf
ANT	Inf	0	1	1	1	5	3	Inf	1	1	1	3	1	5	3	3	1	5
BEE	1	5	0	3	Inf	5	5	1	Inf	3	5	3	1	Inf	5	1	1	1
CER	3	1	Inf	0	5	5	Inf	5	1	5	5	5	Inf	5	5	1	1	5
CHA	1	1	1	1	0	3	Inf	5	5	3	5	Inf	3	1	5	3	5	3
CLU	1	Inf	5	1	5	0	1	1	5	1	3	Inf	1	1	1	1	1	1
EUP	1	5	1	5	1	5	0	5	1	Inf	3	3	3	3	Inf	3	3	5
GNK	5	1	5	1	1	5	1	0	3	1	1	1	3	3	Inf	Inf	3	1
GNT	3	5	1	Inf	Inf	5	3	5	0	5	1	1	5	1	1	3	1	5
KVK	5	5	3	5	5	Inf	3	Inf	1	0	Inf	1	5	1	1	5	5	5
KVM	3	Inf	5	5	3	5	1	3	3	5	0	1	Inf	5	5	1	5	Inf
MOE	3	Inf	1	5	3	3	5	1	5	5	5	0	3	Inf	1	1	3	5
OHL	Inf	1	5	1	5	1	3	Inf	5	1	5	1	0	5	1	3	Inf	1
OOS	3	3	5	Inf	Inf	5	3	1	1	1	1	1	1	0	3	1	Inf	1
STA	5	3	Inf	1	1	3	3	3	Inf	1	3	5	3	1	0	5	Inf	3
STV	Inf	5	1	1	5	Inf	5	5	1	3	Inf	5	1	3	1	0	3	5
WB	5	5	5	5	3	5	1	3	5	Inf	5	1	5	1	5	Inf	0	5
ZWA	3	5	5	1	5	5	Inf	5	Inf	3	5	Inf	5	1	1	5	1	0

Table 3.38: JPL '21 - '22 Cost matrix of a training set

Home/Away	AND	ANT	BEE	CER	CHA	CLU	EUP	GNK	GNT	KVK	KVM	MOE	OHL	OOS	STA	STV	WB	ZWA
AND	0	1	1	1	1	3	2	1	2	2	2	2	2	1	2	1	2	2
ANT	2	0	1	1	1	2	2	2	1	1	1	2	1	2	2	2	1	2
BEE	1	2	0	2	2	2	2	1	2	2	2	2	1	2	2	1	1	1
CER	3	1	2	0	2	3	2	3	1	2	2	2	2	2	2	1	1	3
CHA	1	1	1	1	0	3	3	2	2	2	2	2	2	1	3	2	2	2
CLU	1	2	2	1	2	0	1	1	2	1	2	2	1	1	1	1	1	1
EUP	1	2	1	2	1	3	0	2	1	3	2	2	2	2	2	2	2	2
GNK	2	1	2	1	1	3	1	0	2	1	1	1	2	2	2	2	2	1
GNT	2	3	1	2	2	3	2	2	0	2	1	1	2	1	1	2	1	2
KVK	3	3	2	2	2	3	3	2	1	0	2	1	2	1	1	2	2	2
KVM	2	3	2	2	2	3	1	2	2	3	0	1	2	3	2	1	3	3
MOE	2	2	1	2	2	3	2	1	2	2	2	0	2	2	1	1	2	2
OHL	2	1	2	1	2	1	2	2	2	1	2	1	0	2	1	2	2	1
OOS	3	2	2	2	2	2	2	1	1	1	1	1	1	0	2	1	2	1
STA	2	2	2	1	1	3	3	2	2	1	2	2	2	1	0	2	2	2
STV	2	2	1	1	2	2	3	2	1	2	2	2	1	2	1	0	2	2
WB	2	3	2	3	2	3	1	2	2	2	2	1	2	1	2	2	0	2
ZWA	3	2	3	1	2	3	2	2	2	2	2	2	2	1	1	2	1	0

Table 3.39: JPL '21 - '22 Distance matrix of a training set

There are once again notable differences between the training ranking and the complete ranking (tab. 3.40). However, when comparing these discrepancies to those found in the training ranking for the Schulze method (tab. 3.23), the Floyd-Warshall algorithm provided a better representation in this example. When choosing different training sets, the same observation holds true. Nevertheless, as the aggregate distances remain closely matched, a missing match still has a notable impact on the ranking.

Complete Floyd-Warshall ranking	CLU	AND	ANT	OOS	OHL	GNK	GNT	BEE	STA	STV	CER	EUP	ZWA	KVM	MOE	KVK	CHA	WB
Aggregate distance	22	25	25	25	26	26	27	27	28	28	28	29	29	30	30	31	31	34
Inbound distance	40	30	29	26	28	27	27	25	28	26	25	30	26	29	25	27	26	27
Random training set Floyd-Warshall ranking	CLU	ANT	OOS	OHL	GNK	BEE	AND	STV	GNT	MOE	STA	CHA	EUP	ZWA	CER	WB	KVK	KVM
Aggregate distance	23	26	27	27	27	28	28	30	30	31	32	32	32	33	34	34	34	37
Inbound distance	45	33	27	29	30	28	34	27	28	27	28	29	34	31	26	29	30	30

Table 3.40: JPL '21 - '22 Rankin according to Floyd-Warshall method
Complete set vs random training set

Accuracy

The accuracy was determined by looking at the ability to predict match outcomes based on the Floyd-Warshall ranking. The Distance matrix allows for the calculation of an aggregate distance and an inbound distance. With the aggregate distances, the expected outcome E is

calculated. The home team is expected to win when the ratio remains below 0.5 minus threshold t and vice versa for the away team. If E lies between 0.5 + t and 0.5 - t, a draw is expected.

$$E_{home} = \frac{AggregateDistance_{home} + 1}{AggregateDistance_{home} + AggregateDistanceaway + 2} \tag{3.8}$$

	Baseline (assuming S_home = 1)		**Baseline** (assuming S_home = 0,5)		**Points based ranking**		**Elo Model 4**	
	Accuracy	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*
2020-2021	40,74%	0,2659	23,48%	0,0281	41,63%	0,0293	38,37%	0,0264
2021-2022	42,40%	0,2683	21,85%	0,0328	51,69%	0,0316	53,18%	0,0274
	Elo Model 1		**Schulze method**		**Floyd-Warshall** (Outcome based)		**Floyd-Warshall** (Goal difference based)	
	Accuracy	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*
2020-2021	44,00%	0,0330	42,07%	0,0834	40,30%	0,0318	41,19%	0,0329
2021-2022	54,74%	0,0357	50,34%	0,0616	48,15%	0,0412	48,43%	0,0486

Table 3.41: Floyd-Warshall algorithm vs Schulze method vs Elo rating vs Points based ranking after cross validation

Table 3.41 allows to compare the mechanisms side by side. Although the method based on the goals difference has a slightly higher accuracy than the outcome based method, both Floyd-Warshall algorithms result in a less accurate prediction compared to the initial Points based ranking or every other method.

Allowing for the entire dataset to be used for training, improves the accuracy of a prediction for both seasons (tab. 3.42). As was the case for the Schulze method, using the algorithm in a format with more pairwise comparisons should improve the accuracy without using the benefit of hindsight.

	Floyd-Warshall (Outcome based) 29 folds as training set		**Floyd-Warshall** (Goal difference based) 29 folds as training set		**Floyd-Warshall** (Outcome based) 34 folds as training set		**Floyd-Warshall** (Goal difference based) 34 folds as training set	
	Accuracy	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*	*Accuracy*	*MSE*
2020-2021	40,30%	0,0318	41,19%	0,0329	46,67%	0,0329	44,37%	0,0342
2021-2022	48,15%	0,0412	48,43%	0,0486	52,38%	0,0425	50,36%	0,0503

Table 3.42: Floyd-Warshall algorithm: partial training set vs complete training set after cross validation

Mean Squared Error

The values of the MSE across the Floyd-Warshall algorithm are not as extreme a the ones from the Schulze method, but still higher than for the other mechanisms. Yet, the MSE is still low, suggesting the algorithm could be of some use in scenarios where indirect relationships are an important parameter.

3.4.4 Conclusion Floyd-Warshall algorithm

Implementing the Floyd-Warshall algorithm as a ranking mechanism for a football competition, offers a different perspective compared to the previous methods. By leveraging both direct and indirect metrics, the shortest path between teams is found. In this example, two approaches were studied: outcome-based and goal difference-based. They provide a viable framework, with the latter slightly improving accuracy by interpreting the goal difference as a measure for dominance. But while the algorithm demonstrates potential for handling larger datasets, its performance in this context over 1 season is only moderate compared to the previous mechanisms.

Chapter 4

Conclusion

This paper explores pairwise comparison methods and their role in ranking mechanisms. It aims to integrate theoretical foundations with a practical case study to see how these mechanisms perform in a real-life, non-ideal condition. The paper focusses mainly on the Elo-rating system, the Schulze method and the Floyd-Warshall algorithm. These were evaluated not only separately but also alongside one another in their ability to address complexities of real-life applications.

The practical comparison of ranking mechanisms is focussed around the Belgian Jupiler Pro League. With the match data from the '20-'21 and '21-'22 season, the predictive accuracy, robustness and suitability of various mechanisms was evaluated. Accuracy and mean squared error, MSE, were used to quantify performances, while cross validation techniques were applied to ensure repeatable and consistent comparisons. The numerical results, but more so the understanding of these results, gives an idea of the possibilities and the limitations of each mechanism.

The analysis of the points based ranking was the first indicator to highlight the difficulty in calculating an expected outcome that aligns with the actual outcome. Football is inherently an unpredictable game with countless parameters shifting over the course a season or even a single match. This unpredictability can be extrapolated to virtually any competitive environment.

Although the points based ranking is easy to understand, it fails to account for the varying significance of match results. For example, a mid-tier team defeating a bottom dweller should not weigh equally on a ranking as a victory over a title contender. The Elo rating addresses this discrepancy by taking relative strength between teams into account. And although fine-tuning the parameters proved challenging, its adaptability can be valuable when handling all different kinds of situations.

Compared to the previous two mechanisms, the Schulze method is less transparent but excels in addressing more complex interdependencies by considering both direct and indirect results. While the initial cross validation results were not encouraging, the potential of the Schulze method becomes more evident when applied to larger datasets. Future research is required to draw more definite conclusions, but our results suggest that the Schulze ranking provides a more accurate indication of power dynamics than the previous methods.

Lastly, the Floyd-Warshall algorithm, despite being similar in computational approach, failed to improve on the Schulze method in producing a representative ranking for outcome prediction. Neither of two variations that were tested demonstrated a notable outperformance on the points based or Elo ranking. So although the algorithm's goal to look for the shortest path holds potential, the application in this paper could not succeed in effectively illustrating this potential.

While using the football pitch as a starting point, this paper aims to illustrate the broader applicability of pairwise comparison methods and to underscore the value of understanding their use. By implementing these methods and insights in manager roles, organisations can profit from transparent and deliberate decision making in domains like HR and resource

allocation. In conclusion, this paper should highlight the potential benefits of understanding ranking mechanisms, based on pairwise comparison methods.

A.1 Points based ranking

```
clc; clear; format compact;
warning('off');
%% Explanation
% This code implements a cross-validation approach to evaluate the performance of a
% points-based ranking system for football matches. It uses a predefined number of
% matchdays for training and testing, ensuring chronological consistency in the split
% and thus an equal number of matches for each team.
% 1. Data Loading:
% 2. Cross-Validation Setup:
%    - A cross-validation loop is implemented to iteratively test the model's performance
%      by training on 29 matchdays and testing on 5 consecutive matchdays.
%    - Each fold progresses sequentially
% 3. Training Phase:
%    - For each fold, the training dataset is used to calculate team rankings based on
%      allocated points (3 for a win, 1 for a draw, 0 for a loss).
%    - These rankings serve as the model's basis for predicting match outcomes.
% 4. Testing Phase:
%    - For each match in the test set:
%       - The actual result (normalized based on the goals scored) is compared to the
%         expected result (normalized based on the points ratio).
%       - Predictions are made for home wins, away wins, or draws using a threshold (t).
%         This threshold ensures a reasonable number of predicted draws.
%    - Metrics calculated:
%       - Accuracy: Proportion of correctly predicted outcomes.
%       - MSE: Error between predicted and actual normalized scores.
%       - Draw Counts: Average number of matches predicted as draws.
% 5. Aggregation:
%    - After completing the cross-validation, the mean and standard deviation of accuracy
%      and MSE are computed over all folds and iterations.
%    - Results are printed to provide an overview of model performance.
% 6. Remark:
%    - In this code, the use of iterations is obsolete. However, initially a random seed
%      was used to pick a test set. Unfortunately, this did not ensure an equal number of
%      matches for each team.

%% Load and interpret Result Matrix
dataTable = readtable('D:\Documenten HDD\Matlab\Matches21-22.xlsx');
n_Matchdays = 34; % Total number of matchdays in the season
n_Teams = 18;     % Number of teams in the league
% Preallocate arrays for results
Iterations = 1;
meanAccuracy = zeros(1, Iterations);
stdAccuracy = zeros(1, Iterations);
meanMSE = zeros(1, Iterations);
stdMSE = zeros(1, Iterations);
drawCounts = zeros(1, Iterations);
n_Testfolds = 5;

%% Search loop
for SEARCHMEAN = 1:Iterations
   % Cross-validation loop (leave out 5 matchdays for testing)
   accScores = zeros(n_Matchdays - n_Testfolds, 1);
   MSEScores = zeros(n_Matchdays - n_Testfolds, 1);
   totalDraws = 0;
   for fold = 1:(n_Matchdays - n_Testfolds+1)
       % Select training and test matchdays
       testMatchdays = fold:(fold + (n_Testfolds-1));  % 5 consecutive matchdays for testing
       trainMatchdays = 1:n_Matchdays;
       trainMatchdays(testMatchdays) = [];             % Remaining matchdays for training
       % Establish training and test sets
       trainData = dataTable(ismember(dataTable{:, 1}, trainMatchdays), :); % Training data
       testData = dataTable(ismember(dataTable{:, 1}, testMatchdays), :);   % Testing data
```

```
        % Initialize team points
        points = zeros(n_Teams, 1);
        % Calculate points-based ranking on the training data
        for i = 1:size(trainData, 1)
            home = trainData(i, 2); % Home team index
            away = trainData(i, 3); % Away team index
            diff = trainData(i, 6); % Goal difference
            % Assign points
            if diff > 0 % Home team wins
                points(home) = points(home) + 3;
            elseif diff < 0 % Away team wins
                points(away) = points(away) + 3;
            else % Draw
                points(home) = points(home) + 1;
                points(away) = points(away) + 1;
            end
        end
        % Predict outcomes for the test set
        correctPredictions = 0; % Counter for correct predictions
        squaredError = 0;       % Initialize squared error
        foldDraws = 0;
        for i = 1:size(testData, 1)
            home = testData(i, 2); % Home team index
            away = testData(i, 3); % Away team index
            G_home = testData(i, 4); % Goals scored by home team
            G_away = testData(i, 5); % Goals scored by away team
            % Actual Score (S)
            actualScore = (G_home + 1) / (G_home + G_away + 2);
            % Normalized points ratio
            norm = points(home) / (points(home) + points(away));
            t = 0.025; % Threshold for draws
            % Expected Score
            expectedScore = norm;
            % Update squared error
            squaredError = squaredError + (expectedScore - actualScore)^2;
            % Compare points and predict outcome
            if norm > (0.5 + t) && G_home > G_away
                correctPredictions = correctPredictions + 1; % Home win
            elseif norm < (0.5 - t) && G_home < G_away
                correctPredictions = correctPredictions + 1; % Away win
            elseif abs(norm - 0.5) <= t && G_home == G_away
                correctPredictions = correctPredictions + 1; % Draw
            end
            if abs(norm - 0.5) <= t
                foldDraws = foldDraws + 1;
            end
        end
        % Store results for this fold
        accScores(fold) = correctPredictions / size(testData, 1);  % Accuracy for this fold
        MSEScores(fold) = squaredError / size(testData, 1);        % MSE for this fold
        totalDraws = totalDraws + foldDraws; % Accumulate draw count
    end
    % Aggregate results for this SEARCHMEAN iteration
    meanAccuracy(SEARCHMEAN) = mean(accScores);
    stdAccuracy(SEARCHMEAN) = std(accScores);
    meanMSE(SEARCHMEAN) = mean(MSEScores);
    stdMSE(SEARCHMEAN) = std(MSEScores);
    drawCounts(SEARCHMEAN) = totalDraws / (n_Matchdays - n_Testfolds); % Average draws per fold
end
fprintf('Mean Accuracy over %d iterations: %.2f%%\n', Iterations, mean(meanAccuracy) * 100);
fprintf('MSE over %d iterations: %.4f\n', Iterations, mean(meanMSE));
fprintf('Average expected draws per test-set: %.2f\n', mean(drawCounts));
```

Figure A.1: Matlab code cross validation points based ranking

A.2 Schulze method

```
% Code was tested and approved
% Load and interpret Result Matrix
clc; clear; format compact;
warning('off')
R = [0 12 11;8 0 13;9 7 0];
n = size(R);

%% Create preference matrix G
G = zeros(n);

for i = 1:n
    for j = 1:n
        G(i, j) = R(i, j) - R(j, i);
        if G(i, j) < 0
            G(i, j) = -inf;
        end
    end
end

% The preference graph, represented as matrix:
disp(G);

%% Strongest Path Matrix
% Initialize strongest path
P = G;
for i = 1:n
    for j = 1:n
        if i == j
            P(i, j) = 0; % Draw
        elseif G(i, j) == -inf
            P(i, j) = -inf; % No connection
        end
    end
end

% Schulze Strongest Path Calculation
for k = 1:n
    for i = 1:n
        for j = 1:n
            if i ~= j
                P(i, j) = max(P(i, j), min(P(i, k), P(k, j)));
            end
        end
    end
end

% Strongest Path Matrix P:
disp(P);

%% Calculate Rankings
ranking_scores = zeros(1, n(1,1));
for i = 1:n
    for j = 1:n
        if i ~= j && P(i, j) > P(j, i)
            ranking_scores(i) = ranking_scores(i) + 1;
        end
    end
end

% Sort the ranking scores
[~, ranking] = sort(ranking_scores, 'descend');
% Ranking of options according to the Schulze method:
disp(ranking);
```

Figure A.2: Matlab code example Schulze method (sec. 2.1.4)

```
%% Code was tested and approved
clc; clear; format compact;
warning('off')
%% Load and interpret Result Matrix
dataTable = readtable('/MATLAB Drive/JPL_Matches_Matrix21.csv');
R = dataTable{:,:};
n = size(R);

%% Create preference matrix G
G = zeros(n);
for i = 1:n
    for j = 1:n
        G(i, j) = R(i, j) - R(j, i);
        if G(i, j) < 0
            G(i, j) = -inf;
        end
    end
end
disp('The preference graph, represented as matrix:');
disp(G);

%% Strongest Path Matrix
% Initialize strongest path
P = G;
for i = 1:n
    for j = 1:n
        if i == j
            P(i, j) = 0; % Draw
        elseif G(i, j) == -inf
            P(i, j) = -inf; % No connection
        end
    end
end

% Schulze Strongest Path Calculation
for k = 1:n
    for i = 1:n
        for j = 1:n
            if i ~= j
                P(i, j) = max(P(i, j), min(P(i, k), P(k, j)));
            end
        end
    end
end
disp('Strongest Path Matrix P:');
disp(P);

%% Calculate Rankings with tiebreaker
ranking_scores = zeros(1, n(1,1));
total_path_strengths = zeros(1, n(1,1));

for i = 1:n
    for j = 1:n
        if i ~= j
            % Ranking
            if P(i, j) > P(j, i)
                ranking_scores(i) = ranking_scores(i) + 1;
            end
            % Total path strength
            if P(i, j) > -inf
                total_path_strengths(i) = total_path_strengths(i) + P(i, j);
            end
        end
    end
end

% Ranking
ranking_data = [(1:n)', ranking_scores', total_path_strengths'];
% Sort by dominance score, then by total path strength
ranking_data = sortrows(ranking_data, [-2, -3]);
% Extract final ranking
ranking = ranking_data(:, 1); % Team indices sorted by rank
disp('Ranking of teams with tiebreaker (team, dominance, total strength):');
disp(ranking_data);
```

Figure A.3: Matlab code Schulze method

```
clc; clf; clear; format compact;
warning('off');
%% Load and interpret Result Matrix
R = readtable('D:\Documenten HDD\Matlab\JPL_Matches_Matrix20-21');
R = table2array(R);
n = size(R);
G = zeros(n);
%% Create preference matrix G
for i = 1:n
   for j = 1:n
       G(i, j) = R(i, j) - R(j, i);
       if G(i, j) < 0
           G(i, j) = -inf;
       end
   end
end
%% Initialize adjacency matrix and weights
%   An Edge exists if the goal difference is greater than 0. If both the
%   homa and away round between two teams result in a draw, there will be
%   no preference/edge for either team.
adjacency_matrix = zeros(n);
edge_weights = zeros(n);
for i = 1:n
   for j = 1:n
       if ~isnan(G(i, j)) && G(i, j) > 0
           adjacency_matrix(i, j) = 1;
           edge_weights(i, j) = G(i, j);
       end
   end
end
%% Graph with edge weigths
[source, target] = find(adjacency_matrix); % Get source and target nodes
weights = edge_weights(sub2ind(size(edge_weights), source, target)); % Get
corresponding weights
GRPH = digraph(source, target, weights);
% Plot the graph with edge colors representing weights
team_names = {'AND', 'ANT', 'BEE', 'CER', 'CHA', 'CLU', 'EUP', 'GNK', 'GNT',...
   'KVK', 'KVM', 'MOE', 'OHL', 'OOS', 'STA', 'STV', 'WB', 'ZWA'};
figure;
p = plot(GRPH, 'Layout', 'circle');
title('JPL preference graph 2021-2022');
p.NodeLabel = team_names;               % Insert team names
colormap jet;
p.EdgeCData = GRPH.Edges.Weight;        % Assign weights to edge colors
colorbar;
p.LineWidth = GRPH.Edges.Weight;
```

Figure A.4: Matlab code plotting preference graph

```
clc; clear; format compact;
%% Read the data from the Excel file
dataTable = readtable('D:\\Documenten HDD\\Matlab\\Matches20-21.xlsx');
matchdayIndex = dataTable{:, 1};    % Matchday index
homeTeamIndex = dataTable{:, 2};    % Home team index
awayTeamIndex = dataTable{:, 3};    % Away team index
scoreDifference = dataTable{:, 6};  % Delta goals
n_Matchdays = 34;
n_Teams = 18;
n_Testfolds = 5;
t = 0.025;
% Preallocate arrays for results
accScores = zeros(n_Matchdays - n_Testfolds, 1);
MSEScores = zeros(n_Matchdays - n_Testfolds, 1);
totalDraws = 0;
%% Cross-validation loop (leave out 5 matchdays for testing)
for fold = 1:(n_Matchdays - n_Testfolds + 1)
   % Select training and test matchdays
   testMatchdays = fold:(fold + (n_Testfolds - 1));  % 5 consecutive matchdays for testing
   trainMatchdays = setdiff(1:n_Matchdays, testMatchdays);
   trainData = dataTable(ismember(dataTable{:, 1}, trainMatchdays), :); % Training data
   testData = dataTable(ismember(dataTable{:, 1}, testMatchdays), :);   % Testing data
   %% Pairwise preference matrix
   prefMatrix = zeros(n_Teams, n_Teams);
   for i = 1:size(trainData, 1)
       home = trainData{i, 2};    % Home team index
       away = trainData{i, 3};    % Away team index
       Delta = trainData{i, 6};   % Delta goals
       prefMatrix(home, away) = prefMatrix(home, away) + Delta;
       %prefMatrix(away, home) = prefMatrix(away, home) - Delta;
   end
   %% Pairwise preference graph
   G = zeros(n_Teams, n_Teams);
   for i = 1:n_Teams
       for j = 1:n_Teams
           G(i, j) = prefMatrix(i, j) - prefMatrix(j, i);
           if G(i, j) < 0
               G(i, j) = -inf;
           end
       end
   end
   %% Strongest Path Matrix
   P = G;
   P(eye(n_Teams) == 1) = 0; % Set diagonal to -inf
   for k = 1:n_Teams
       for i = 1:n_Teams
           for j = 1:n_Teams
               if i ~= j
                   P(i, j) = max(P(i, j), min(P(i, k), P(k, j)));
               end
           end
       end
   end
   %% Calculate Rankings with Tiebreaker
   ranking_scores = zeros(1, n_Teams);
   total_path_strengths = zeros(1, n_Teams);
   for i = 1:n_Teams
       for j = 1:n_Teams
           if i ~= j
               if P(i, j) > P(j, i)
                   ranking_scores(i) = ranking_scores(i) + 1;
               end
```

```
                total_path_strengths(i) = total_path_strengths(i) + max(0, P(i, j));
            end
        end
    end
    ranking_data = [(1:n_Teams)', ranking_scores', total_path_strengths'];
    ranking_data = sortrows(ranking_data, [-2, -3]);
    ranking = ranking_data(:, 1);
    %% Predict outcomes for the test set
    correctPredictions = 0; % Counter for correct predictions
    squaredError = 0;       % Initialize squared error
    foldDraws = 0;
    for i = 1:size(testData, 1)
        home = testData(i, 2);      % Home team index
        away = testData(i, 3);      % Away team index
        G_home = testData(i, 4);    % Home goals
        G_away = testData(i, 5);    % Away goals
        Delta = testData(i, 6);     % Delta goals
        % Normalise expected outcome and actual outcome
        S_home = (G_home + 1) / (G_home + G_away + 2);
        E_Dom = ((ranking_scores(home)+1)/(ranking_scores(home)+ranking_scores(away)+2));
        E_TPS =
(total_path_strengths(home)+1)/((total_path_strengths(home)+total_path_strengths(away)+2));
        % Predict winner based on dominance score an total path strength
        if ranking_scores(home) > ranking_scores(away) && Delta > 0
            correctPredictions = correctPredictions + 1;        % Correct home win
        elseif ranking_scores(home) < ranking_scores(away) && Delta < 0
             correctPredictions = correctPredictions + 1;       % Correct away win
        elseif ranking_scores(home) == ranking_scores(away)
            if E_TPS > (0.5 + t) && Delta > 0
                correctPredictions = correctPredictions + 1;    % Correct home win
            elseif E_TPS < (0.5- t) && Delta < 0
                correctPredictions = correctPredictions + 1;    % Correct away win
            elseif abs(E_TPS- 0.5) <= t && Delta == 0
                correctPredictions = correctPredictions + 1;    % Correct draw prediction
            end
            if abs(E_TPS - 0.5) <= t
            foldDraws = foldDraws + 1;
            end
        end
        if ranking_scores(home) ~= ranking_scores(away)
            squaredError = squaredError + (S_home - E_Dom)^2;
        else
            squaredError = squaredError + (S_home - E_TPS)^2;
        end
        % % Predict winner based on rankings, tie-braker present, no draws predicted
        % if find(ranking == home) < find(ranking == away) && Delta > 0
        %     correctPredictions = correctPredictions + 1; % Correct home win
        % elseif find(ranking == home) > find(ranking == away) && Delta < 0
        %     correctPredictions = correctPredictions + 1; % Correct away win
        % end
        % squaredError = squaredError + (S_home - E_Dom)^2;
    end
    % Store results for this fold
    accScores(fold) = correctPredictions / size(testData, 1);   % Accuracy for this fold
    MSEScores(fold) = squaredError / size(testData, 1);         % MSE for this fold
    totalDraws = totalDraws + foldDraws;                        % Accumulate draw count
end
meanAccuracy = mean(accScores);
stdAccuracy = std(accScores);
meanMSE = mean(MSEScores);
stdMSE = std(MSEScores);
avgDraws = totalDraws / (n_Matchdays - n_Testfolds); % Average draws per fold
```

Figure A.5: Matlab code cross validation Schulze method based ranking

A.3 Floyd-Warshall algorithm

```
clc; clear; format compact;
warning('off')
%% Load and interpret Result Matrix
dataTable = readtable('D:\Documenten HDD\Matlab\JPL_Matches_Matrix21-22');
R = dataTable{:,:};
n = size(R);
disp('The pairwise preference matrix of the JPL ''21 - ''22 season:');
disp(R);
nTeams = n(1,1);
Cost = inf(nTeams);
%% Populate the Cost matrix
% The cost is calculated based on the goal difference. A goal difference of
% 1 goal will result in the same costs as previously used [1 for a win, 3
% for a draw and 5 for a loss]. A larger goal difference will have a larger
% impact.
for i = 1:nTeams
   for j = 1:nTeams
       if i ~= j
           if ~isnan(R(i, j))
               if R(i, j) > 0          % Home win
                   Cost(i, j) = 2*(1 / (R(i, j) + 1));
               elseif R(i, j) == 0     % Draw
                   Cost(i, j) = 3;
               else                    % Away win
                   Cost(i, j) = 2.5*(-R(i, j) + 1);
               end
           end
       else
           Cost(i, j) = inf;
       end
   end
end
disp('The cost matrix:');
disp(R);
%% Floyd-Warshall Algorithm (Minimization)
Delta = Cost;
for k = 1:nTeams
   for i = 1:nTeams
       for j = 1:nTeams
           if Delta(i, j) > Delta(i, k) + Delta(k, j)
               Delta(i, j) = Delta(i, k) + Delta(k, j);
           end
       end
   end
end
for i = 1:nTeams
   Delta(i, i) = 0;    % Ensure diagonal remains consistent
end
disp('Final Distance Matrix (Delta):');
disp(Delta);
%% Compute team rankings based on aggregate performance
aggregateScores = sum(Delta, 2);    % Row-wise sum for aggregate scores
inboundScores = sum(Delta, 1)';     % Column-wise sum for inbound scores
resultsMatrix = [(1:nTeams)', aggregateScores, inboundScores];
sortedResults = sortrows(resultsMatrix, [2, -3]); % Sort by aggregate, then inbound score
% Display results
disp('Results Matrix (Sorted):');
disp('Column 1: Team | Column 2: Aggregate Distance | Column 3: Inbound Score');
disp(sortedResults);
```

Figure A.6: Matlab code Floyd-Warshall algorithm

```
clc; clear; format compact;
%% Read the data from the Excel file
dataTable = readtable('D:\Documenten HDD\Matlab\Matches21-22.xlsx');
matchdayIndex = dataTable{:, 1};    % Matchday index
homeTeamIndex = dataTable{:, 2};    % Home team index
awayTeamIndex = dataTable{:, 3};    % Away team index
scoreDifference = dataTable{:, 6};  % Delta goals
n_Matchdays = 34;
n_Teams = 18;
n_Testfolds = 5;
t = 0.008;
% Preallocate arrays for results
accScores = zeros(n_Matchdays - n_Testfolds, 1);
MSEScores = zeros(n_Matchdays - n_Testfolds, 1);
totalDraws = 0;
%% Cross-validation loop
for fold = 1:(n_Matchdays - n_Testfolds + 1)
   % Split data into training and test sets
   testMatchdays = fold:(fold + (n_Testfolds - 1));
   trainMatchdays = setdiff(1:n_Matchdays, testMatchdays);
   trainData = dataTable(ismember(dataTable{:, 1}, trainMatchdays), :);
   testData = dataTable(ismember(dataTable{:, 1}, testMatchdays), :);
   %% Training Cost matrix
   Cost = inf(n_Teams, n_Teams); % Initialize cost matrix with infinity
   for i = 1:size(trainData, 1)
       home = trainData{i, 2};
       away = trainData{i, 3};
       Delta = trainData{i, 6}; % Goal difference
       %Assign costs based on match outcome
       if Delta > 0
           Cost(home, away) = 1;% Home win
       elseif Delta == 0
           Cost(home, away) = 3; % Draw
       else
           Cost(home, away) = 5;% Away win
       end
       % Assign costs based on actual goal difference
       % if Delta > 0
       %     Cost(home, away) = 2 / (Delta + 1); % Home win
       % elseif Delta == 0
       %     Cost(home, away) = 3; % Draw
       % else
       %     Cost(home, away) = 2.5 * (-Delta + 1); % Away win
       % end
   end
   Cost(eye(n_Teams) == 1) = 0; % Set diagonal to zero (self-loops)
   %% Floyd-Warshall Algorithm
   Delta = Cost;
   for k = 1:n_Teams
       for i = 1:n_Teams
           for j = 1:n_Teams
               Delta(i, j) = min(Delta(i, j), Delta(i, k) + Delta(k, j));
           end
       end
   end
   %% Rankings based on shortest path distances
   aggregateScores = sum(Delta, 2);    % Row-wise sum (aggregate performance)
   inboundScores = sum(Delta, 1)';     % Column-wise sum (inbound performance)
   ranking_data = [(1:n_Teams)', aggregateScores, inboundScores];
   ranking_data = sortrows(ranking_data, [2, 3]);
   ranking = ranking_data(:, 1);
   %% Predict outcomes for the test set
```

```
    correctPredictions = 0;
    squaredError = 0;
   foldDraws = 0;
    for i = 1:size(testData, 1)
        home = testData(i, 2);
        away = testData(i, 3);
        G_home = testData(i, 4);
        G_away = testData(i, 5);
        Delta_actual = testData(i, 6);
        % Normalise expected and actual outcome
        S_home = (G_home + 1) / (G_home + G_away +2);
        E_home = (aggregateScores(home)+1)/(aggregateScores(home)+aggregateScores(away)+2);
        % Prediction based on rankings
        if E_home < (0.5-t) && Delta_actual > 0
            correctPredictions = correctPredictions + 1; % Correct home win
        elseif E_home > (0.5+t) && Delta_actual < 0
            correctPredictions = correctPredictions + 1; % Correct away win
        elseif abs(E_home - 0.5) <= t && Delta_actual == 0
            correctPredictions = correctPredictions + 1; % Correct draw
        end
        if abs(E_home - 0.5) <= t
            foldDraws = foldDraws + 1;
        end
        % Compute squared error
        squaredError = squaredError + (E_home - S_home)^2;
    end
    % Store results for this fold
    accScores(fold) = correctPredictions / size(testData, 1);
    MSEScores(fold) = squaredError / size(testData, 1);
    totalDraws = totalDraws + foldDraws;                        % Accumulate draw count
end
% Final results
meanAccuracy = mean(accScores);
stdAccuracy = std(accScores);
meanMSE = mean(MSEScores);
stdMSE = std(MSEScores);
avgDraws = totalDraws / (n_Matchdays - n_Testfolds); % Average draws per fold
% Display results
fprintf('Mean Accuracy: %.2f%%\n', meanAccuracy * 100);
fprintf('Mean MSE: %.4f\n', meanMSE);
fprintf('Average expected draws per test-set: %.2f\n', avgDraws);
```

Figure A.7: Matlab code cross validation Floyd-Warshall algorithm based ranking

A.4 Elo rating

```
% Code was tested and approved
clc;clear
%% Input
n = 18;                         % Number of teams
m = (34*n)/2;                   % Total matches
match_results = zeros(m, 3);    % [Home Team, Away Team, Result]
dataTable = readtable('/MATLAB Drive/JPL_Matches.xlsx');
dataTable = table2array(dataTable);

% Fill matrix for Home Team, Away Team and Goal difference
match_results(:,1) = dataTable(:,2);    % Home team
match_results(:,2) = dataTable(:,3);    % Away team
match_results(:,3) = dataTable(:,6);    % Goal difference

% Fill matrix for Home Team, Away Team and Result
match_results(match_results(:, 3) > 0, 3) = 1;    % Positive values become 1
match_results(match_results(:, 3) == 0, 3) = 0.5; % Zero values become 0.5
match_results(match_results(:, 3) < 0, 3) = 0;    % Negative values become 0

% Elo parameters
K = 30;                 % Constant K factor
R_0 = 1500;             % Initial rating for all teams
R = R_0 * ones(1, n);   % Initialise rating matrix

%% Elo Calculation
for i = 1:m
    home_team = match_results(i, 1);
    away_team = match_results(i, 2);
    result = match_results(i, 3);

    % Current ratings
    R_home = R(home_team);
    R_away = R(away_team);

    % Expected scores
    E_home = 1 / (1 + 10^((R_away - R_home) / 400));
    E_away = 1 / (1 + 10^((R_home - R_away) / 400));

    % Update ratings
    if result == 1  % Home win
        S_home = 1; S_away = 0;
    elseif result == 0.5  % Draw
        S_home = 0.5; S_away = 0.5;
    else  % Away win
        S_home = 0; S_away = 1;
    end

    R(home_team) = R_home + K * (S_home - E_home);
    R(away_team) = R_away + K * (S_away - E_away);
end

[sortedRatings, originalIndices] = sort(R, 'descend');
result = [originalIndices; sortedRatings]
```

Figure A.8: Matlab code Elo Rating JPL Basic

```
% Code was tested and approved
clc; clear;clf

%% Input
n = 18;                         % Number of teams
m = (34*n)/2;                   % Total matches
match_results = zeros(m, 4);    % [Home Team, Away Team, Result]
dataTable = readtable('/MATLAB Drive/JPL_Matches.xlsx');
dataTable = table2array(dataTable);

% Goal difference G
G_home = dataTable(:,4);   % Goals home team
G_away = dataTable(:,5);   % Goals away team

% Actual results S
S_home = (G_home+1) ./ (G_home+G_away+2);
S_away = (G_away+1) ./ (G_away+G_home+2);

% Fill matrix for Home Team, Away Team and Goal difference
match_results(:,1) = dataTable(:,2);    % Home team
match_results(:,2) = dataTable(:,3);    % Away team

% Elo parameters
R_0 = 1500;                 % Initial rating for all teams
MSE_optimal = Inf;          % Initialize the best MSE
K_optimal = 0;              % Initialize the best K
K = 5:0.1:100;
MSE = zeros(size(K));

%% Loop through K values
for idx = 1:length(K)
    K = K(idx);             % Current K value
    R = R_0 * ones(1, n);   % Initialise rating matrix for current K
    MSE = zeros(1, m);      % Initialize MSE for each match

    %% Elo Calculation
    for i = 1:m
        home_team = match_results(i, 1);
        away_team = match_results(i, 2);

        % Current ratings
        R_home = R(home_team);
        R_away = R(away_team);

        % Expected scores
        E_home = 1 / (1 + 10^((R_away - R_home) / 400));
        E_away = 1 / (1 + 10^((R_home - R_away) / 400));

        % Update ratings
        R(home_team) = R_home + K .* (S_home(i) - E_home);
        R(away_team) = R_away + K .* (S_away(i) - E_away);

        % Mean Squared Error
        MSE(i) = (E_home - S_home(i)).^2 + (E_away - S_away(i)).^2;
    end

    % Average MSE for current K
    MSE_avg = mean(MSE);
    MSE(idx) = MSE_avg;

    % Update best K if current MSE is lower
    if MSE_avg < MSE_optimal
        MSE_optimal = MSE_avg;
        K_optimal = K;
    end
end

%% Plot
figure;
plot(K, MSE, 'LineWidth', 1.5); xlabel('Adjustment factor, K'); ylabel('Mean Squared Error, MSE');
title('MSE in function of K');
grid on;

fprintf('Optimal K: %.1f\n', K_optimal); fprintf('Lowest MSE: %.4f\n', MSE_optimal);
```

Figure A.9: Matlab code fine-tune adjustment factor K

```
% Code was tested and approved
clc; clear; tic
%% Input
n = 18;                          % Number of teams
m = (34*n)/2;                    % Total matches
match_results = zeros(m, 4);     % [Home Team, Away Team, Result]
dataTable = readtable('/MATLAB Drive/JPL_Matches.xlsx');
dataTable = table2array(dataTable);
% Goal difference G
G_home = dataTable(:,4);   % Goals home team
G_away = dataTable(:,5);   % Goals away team
% Actual results S
S_home = (G_home+1) ./ (G_home+G_away+2);
S_away = (G_away+1) ./ (G_away+G_home+2);
% Fill matrix for Home Team, Away Team and Goal difference
match_results(:,1) = dataTable(:,2);    % Home team
match_results(:,2) = dataTable(:,3);    % Away team
% Elo parameters
R_0 = 1500;                % Initial rating for all teams
best_MSE = Inf;            % Initialize the best MSE
best_K = 0;                % Initialize the best K
best_h = 0;                % Initialize the best h
% Arrays for tracking
K_values = 1:0.1:40;
h_values = 1:0.1:40;
MSE_matrix = zeros(length(K_values), length(h_values)); % Store MSE values
%% Loop through K and h values
for K_idx = 1:length(K_values)
   K = K_values(K_idx);
   for h_idx = 1:length(h_values)
       h = h_values(h_idx);
       R = R_0 * ones(1, n);   % Initialize rating matrix for current K and h
       MSE = zeros(1, m);      % Initialize MSE for each match
       %% Elo Calculation
       for i = 1:m
           home_team = match_results(i, 1);
           away_team = match_results(i, 2);
           % Current ratings
           R_home = R(home_team);
           R_away = R(away_team);
           % Expected scores with home advantage
           E_home = 1 / (1 + 10^((R_away - R_home - h) / 400));
           E_away = 1 / (1 + 10^((R_home - R_away + h) / 400));
           % Update ratings
           R(home_team) = R_home + K .* (S_home(i) - E_home);
           R(away_team) = R_away + K .* (S_away(i) - E_away);
           % Mean Squared Error (MSE)
           MSE(i) = (E_home - S_home(i)).^2 + (E_away - S_away(i)).^2;
       end
       % Average MSE for current K and h
       avg_MSE = mean(MSE);
       MSE_matrix(K_idx, h_idx) = avg_MSE; % Store MSE
       % Update best K and h if current MSE is lower
       if avg_MSE < best_MSE
           best_MSE = avg_MSE;
           best_K = K;
           best_h = h;
       end
   end
end
%% Plot Results
figure;
numLevels = 50; % Number of contour levels for finer detail
contour(h_values, K_values, MSE_matrix, numLevels, 'LineWidth', 1.2);
xlabel('Home Advantage, h');
ylabel('Adjustment factor, K');
title('MSE in function of K and h');
colorbar;
grid on;
%% Output Results
best_K;best_h;best_MSE;toc
```

Figure A.10: Matlab code fine-tune adjustment factor K & home advantage h

```
clc; clear; format compact;
warning('off');
%% Explanation
% This code implements a cross-validation approach to evaluate the performance of an
% Elo rating based ranking system for football matches. It uses a predefined number of
% matchdays for training and testing, ensuring chronological consistency in the split
% and thus an equal number of matches for each team.
% 1. Data Loading:
% 2. Elo Rating System:
% 3. Cross-Validation Setup:
%    - A cross-validation loop is implemented to iteratively test the model's performance
%      by training on 29 matchdays and testing on 5 consecutive matchdays.
%    - Each fold progresses sequentially
% 4. Training Phase:
%    - Elo ratings are updated for all teams based on the results in training set.
%    - The actual and expected outcomes of matches are used to refine team ratings.
% 5. Testing Phase:
%    - Predictions are made for matches in the test set using Elo ratings from the training
set.
%    - Metrics are calculated:
%      - Accuracy: Proportion of correctly predicted outcomes (home win, away win, draw).
%      - MSE: Difference between the predicted and actual normalized scores.
%      - Draw count: Average number of matches predicted as draws.
% 6. Evaluation:
%    - Results are averaged across folds to compute:
%      - Mean and standard deviation of accuracy and MSE.
%      - Average predicted draws per fold.

%% Load and interpret Result Matrix
dataTable = readtable('D:\Documenten HDD\Matlab\Matches21-22.xlsx'); % Load the data
n_Matchdays = 34; % Total number of matchdays in the season
n_Teams = 18;     % Number of teams in the league
% Elo rating parameters
K = 19.3;   % Adjustment factor
h = 6.9;    % Home advantage
R_0 = 1500; % Initial Elo rating for all teams
% Preallocate arrays for results
n_Testfolds = 5;
accScores = zeros(n_Matchdays - n_Testfolds, 1);
MSEScores = zeros(n_Matchdays - n_Testfolds, 1);
totalDraws = 0;
%% Cross-validation loop (leave out 5 matchdays for testing)
for fold = 1:(n_Matchdays - n_Testfolds+1)
   % Select training and test matchdays
   testMatchdays = fold:(fold + (n_Testfolds-1));  % 5 consecutive matchdays for testing
   trainMatchdays = 1:n_Matchdays;
   trainMatchdays(testMatchdays) = [];             % Remaining matchdays for training
   % Establish training and test sets
   trainData = dataTable(ismember(dataTable{:, 1}, trainMatchdays), :); % Training data
   testData = dataTable(ismember(dataTable{:, 1}, testMatchdays), :);   % Testing data
   % Initialize Elo ratings for all teams
   R = R_0 * ones(n_Teams, 1);
   % Update Elo ratings based on training data
   for i = 1:size(trainData, 1)
       home = trainData{i, 2};     % Home team index
       away = trainData{i, 3};     % Away team index
       G_home = trainData{i, 4};   % Goals scored by home team
       G_away = trainData{i, 5};   % Goals scored by away team

       % Calculate actual outcome
       S_home = (G_home+1) ./ (G_home+G_away+2);
       S_away = (G_away+1) ./ (G_away+G_home+2);
```

```
        % Expected outcome based on Elo ratings
        E_home = 1 / (1 + 10^((R(away) - R(home) - h) / 400));
        E_away = 1 / (1 + 10^((R(home) - R(away) + h) / 400));
        % Update Elo ratings
        R(home) = R(home) + K * (S_home - E_home);
        R(away) = R(away) + K * (S_away - E_away);
    end
    % Predict outcomes for the test set
    correctPredictions = 0; % Counter for correct predictions
    squaredError = 0;       % Initialize squared error
    foldDraws = 0;
    for i = 1:size(testData, 1)
        home = testData(i, 2); % Home team index
        away = testData(i, 3); % Away team index
        G_home = testData(i, 4); % Goals scored by home team
        G_away = testData(i, 5); % Goals scored by away team
        % Actual Score (S)
        S_home = (G_home + 1) / (G_home + G_away + 2);
        % Expected outcome based on Elo ratings
        E_home = 1 / (1 + 10^((R(away) - R(home) - h) / 400));
        E_away = 1 - E_home;
        % Update squared error
        squaredError = squaredError + (E_home - S_home)^2;
        % Predict match outcome
        if E_home > (0.5 + 0.025) && G_home > G_away
            correctPredictions = correctPredictions + 1; % Home win
        elseif E_home < (0.5 - 0.025) && G_home < G_away
            correctPredictions = correctPredictions + 1; % Away win
        elseif abs(E_home - 0.5) <= 0.025 && G_home == G_away
            correctPredictions = correctPredictions + 1; % Draw
        end
        if abs(E_home - 0.5) <= 0.025
            foldDraws = foldDraws + 1;
        end
    end
    % Store results for this fold
    accScores(fold) = correctPredictions / size(testData, 1);   % Accuracy for this fold
    MSEScores(fold) = squaredError / size(testData, 1);         % MSE for this fold
    totalDraws = totalDraws + foldDraws;                        % Accumulate draw count
end
% Aggregate results
meanAccuracy = mean(accScores);
stdAccuracy = std(accScores);
meanMSE = mean(MSEScores);
stdMSE = std(MSEScores);
avgDraws = totalDraws / (n_Matchdays - n_Testfolds); % Average draws per fold
% Final results
fprintf('Mean Accuracy: %.2f%%\n', meanAccuracy * 100);
fprintf('Mean MSE: %.4f\n', meanMSE);
fprintf('Average expected draws per test-set: %.2f\n', avgDraws);
```

Figure A.11: Matlab code cross validation Elo rating based ranking

Graphs

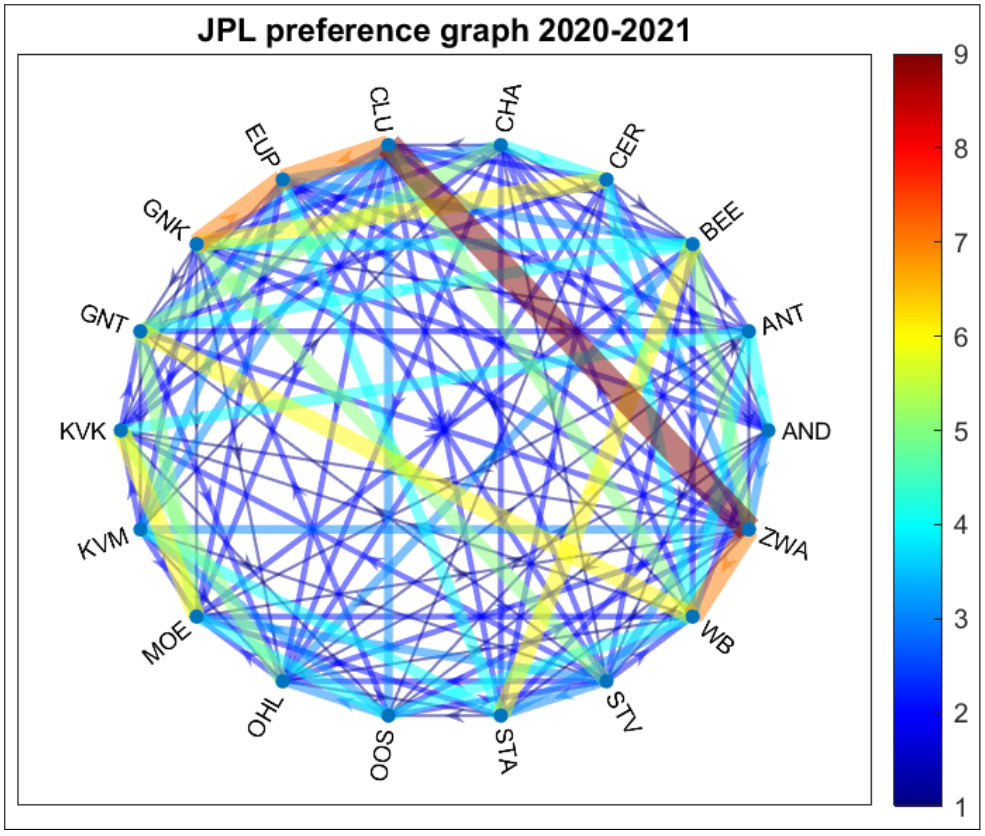

Figure B.1: JPL '20 - '21 Preference graph

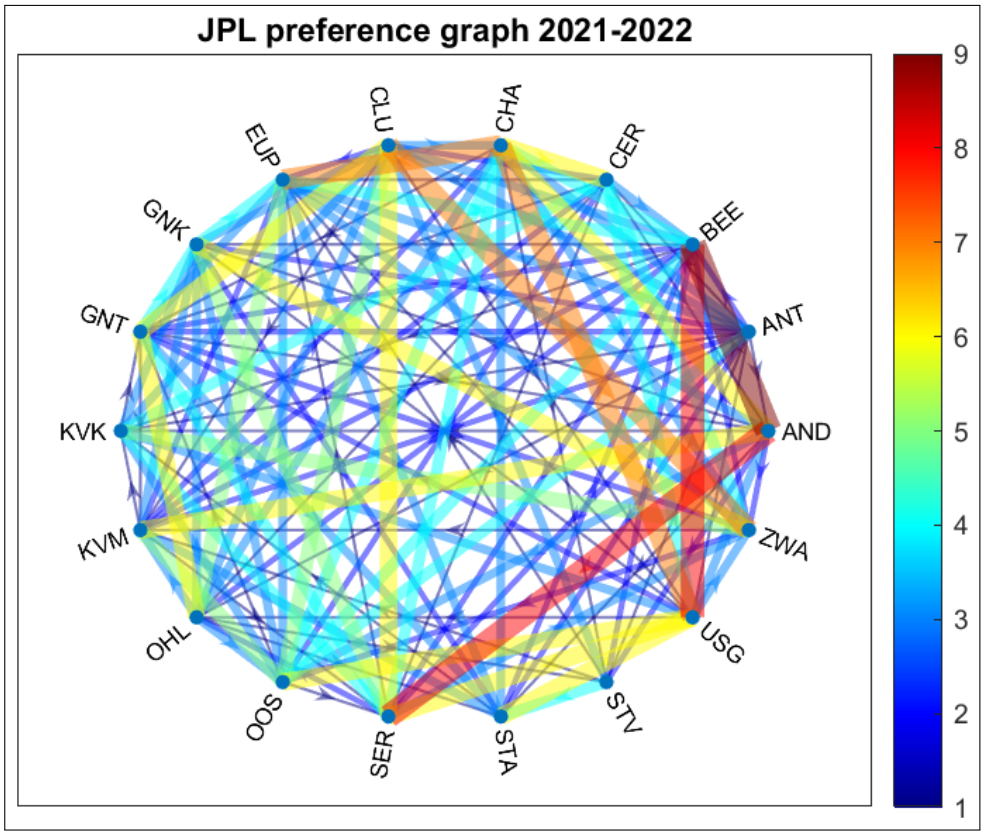

Figure B.2: JPL '21 - '22 Preference graph

Bibliography

[1] *2009 Burlington mayoral election*. URL: https://en.wikipedia.org/wiki/2009_Burlington_mayoral_election (visited on 11/27/2024).

[2] S. J. Brams and D. R. Herschbach. 'The Science of Elections'. In: *Science* 292.5521 (2001).

[3] *Center squeeze*. URL: https://en.wikipedia.org/wiki/Center_squeeze (visited on 12/08/2024).

[4] *Condorcet winner criterion*. URL: https://en.wikipedia.org/wiki/Condorcet_winner_criterion (visited on 11/30/2024).

[5] *Condorcet winner criterion: Examples of failure*. URL: https://en.wikipedia.org/wiki/Condorcet_winner_criterion (visited on 11/12/2024).

[6] *Cross Validation (statistics)*. URL: https://en.wikipedia.org/wiki/Cross-validation_(statistics) (visited on 12/27/2024).

[7] *Duvager's law – more guidelines than actual rules?* URL: https://www.electoral-reform.org.uk/duvagers-law-more-guidelines-than-actual-rules/ (visited on 11/30/2024).

[8] *Eerste klasse A 2020-21 (voetbal België)*. URL: https://nl.wikipedia.org/wiki/Eerste_klasse_A_2020-21_(voetbal_Belgi%C3%AB) (visited on 11/01/2024).

[9] *Eerste klasse A 2021-22 (voetbal België)*. URL: https://nl.wikipedia.org/wiki/Eerste_klasse_A_2021-22_(voetbal_Belgi%C3%AB) (visited on 11/01/2024).

[10] A. Elo. 'The Rating of Chessplayers, Past and Present'. In: (2008).

[11] M. S. F. Fouss M. Saerens. *Algorithms and Models for Network Data and Link Analysis*. 2024.

[12] *FIFA Women's World Ranking*. URL: https://en.wikipedia.org/wiki/FIFA_Women%27s_World_Ranking (visited on 12/01/2024).

[13] T. Gérard. *Le classement Elo et les "expected goals" comme indicateurs de performance lors de la Coupe du Monde 2018 de football*. 2019.

[14] *Independence of irrelevant alternatives*. URL: https://en.wikipedia.org/wiki/Independence_of_irrelevant_alternatives (visited on 11/30/2024).

[15] A. Langville and C. Meyer. *Who's #1?: The Science of Rating and Ranking*. Princeton University Press, 2012.

[16] P. Louridas. *Real–World Algorithms: A Beginner's Guide*. MIT Press, 2017. ISBN: 9780262035705.

[17] *Majority winner criterion*. URL: https://en.wikipedia.org/wiki/Majority_winner_criterion (visited on 11/30/2024).

[18] M. Saerens et al. 'Randomized Shortest-Path Problems: Two Related Models'. In: *Neural Computation* 21.8 (2009), pp. 2363–2404.

[19] *Schulze method: Satisfied and failed criteria*. URL: https://en.wikipedia.org/wiki/Schulze_method (visited on 11/18/2024).

[20] A. Sen. 'The Formulation of Rational Choice'. In: *The American Economic Review* (1994).

[21] *Smith Criterion*. URL: https://en.wikipedia.org/wiki/Smith_set (visited on 11/30/2024).

[22] D. R. Woodall. 'Properties of Preferential Election Rules'. In: *Voting matters* (1994).