Summary Sheet

This paper explores optimal team selection and schedule creation for an international team sports tournament, using selective annealing.

In the first section of *Problem 1*, we chose association football as our team sport. In the second section of problem one, we first assigned a certain number of teams that will play in the GSL to each continent, and then selected that number of national teams from each continent. To choose the number of teams assigned to each continent, we pinpointed 2 measurable factors that represent the geographic diversity and competitive equity in our team selection; (1) **population**, and (2) **team quality**. Then, we employed a **piecewise linear function** to assign a number of teams to each continent. This function ensured that each continent had at least 2 teams (as required by the question), and accounted for both population and team quality. Following this, we selected the best teams from each continent, using FIFA rankings. We chose from best to worse, until the number of teams allocated to each continent had been chosen.

In *Problem 2*, we first pinpointed the essential assessment factors for the most optimal schedule: distance travelled, breaks (i.e. consecutive home/away games), carry over effect (COE), and disparity in each of these factors between teams. Each of these factors should be minimised to maximise the geographic diversity, competitive equity, environmental friendliness, economic sustainability and logistical feasibility of our schedule. Then, we selected the mirrored double round robin (M2RR) structure for our schedule. This structure provides a fair distribution of games across teams and balanced competitiveness of matchups - each team plays all other teams twice, in home and away venues. Additionally, we provided every team 4 byes, to allow our model to optimise COE (unweighted COE value of a compact M2RR is fixed). This resulted in a tournament consisting of 42 rounds, with 9-10 matches per round.

To find the optimal matches, we employed **selective annealing**. Our **objective function** penalised distance, breaks, COE, as well as the disparity in each of these factors between teams. Following this, we created our base schedule using the **Berger Tables method**, where ghost teams were created to represent byes (this schedule does not optimise distance travelled or COE). Then, using our selective annealing program, we applied **random perturbations** to the base schedule to find the matches with the lowest penalty. Finally, the days of play were calculated. Taking into account variables such as **popularity of weekdays** (in terms of stadium attendance and online viewership), changes in **player condition** based on time of day and rest difference, two approaches were appraised: the first kept all games at the same local time for each country, whilst the second staggered games. We ultimately chose the first approach for equity of rest difference, and thus calculated the cycle of game days.

In *Problem 3*, we employed the same methodology as in Problem 1 to select the 24 teams. Then, using the Berger Tables method, we created a base schedule for 24 teams (byes were represented by 2 ghost teams). We inputted this base team into our selective annealing program, which was adjusted for 24 teams. This produced an expanded schedule for 24 teams.

In *Problem 4*, we discussed the **applications of our current model**, as well as methods to generalise our model to encompass **all team sports**. Our current model is applicable to almost all internationally recognised sports. We generalised our model to include sports without an international governing body and ranking system by creating regional selection tournaments. Further, we included sports with less than 20 national teams by removing team selection and adjusting the number of teams in our model. We also included sports with multi-day matches (notably test cricket) by adjusting the number of teams accepted into the GSL, and team racing sports by simplifying our model to account only for travel distance.

Letter to the Decision Makers

Dear esteemed Decision Makers of the IMMC,

Our team has arrived at a recommendation for the optimal Global Sports League schedule. Our schedule-making methodology is a holistic, resource-efficient approach to this problem, which generates a schedule that is equitable, and also ensures the GSL's geographic diversity, environmental and economic sustainability and logistical feasibility.

Firstly, we mathematically justified our 20 team selection, considering international rankings and populations to ensure that each continent had an appropriate, mathematically justifiable number of representatives. The methodology of this section broadly applicable; it is easily scaled to any number of teams, and can utilise any form of ranking system of national teams within the sport.

The problem has given us four categories to sort factors affecting the strength of a schedule into: **competitive equity**, **logistical feasability**, **economic and environmental sustainability** and **geographic diversity**. Within these categories, we ascertained quantifiable factors, and determined their weighting in relation with one another. These are **distance travelled**, **breaks**, and **carry over effect**. We have formulated a model that, utilising our simulated annealing program, accurately weighs and accounts for these factors and ultimately produces the optimal schedule.

Once these matchups were calculated, the cycle of days on which games would be played was determined. We attempted to maximise competitive equality as well as profits. With both the matchups and the days of play, the complete schedule could be produced.

Our model is **widely applicable**, and **easily generalised**. Our original model is valid for almost all internationally recognised sports that meet the GSL's definition of a team sport, including cricket, hockey, volleyball, basketball, baseball, rugby, Australian rules football, netball, softball, lacrosse, and polo. We have further generalised the model so that it can be applied to any team sport, even those not meeting the GSL's criteria for a team sport. This includes sports with limited global reach, sports with matches played over multiple days and racing sports.

We thank you for your confidence in our team, and hope that our schedule serves as an effective and efficient solution for the Global Sports League.

Kind regards,

Team #1000110649 Sydney, Australia



GSL Schedule (01/01-04/08)



In the top half, a gold border around a country's flag represents a home game. In the bottom half, a gold border around a country's flag represents an away game. All times and dates are in local time .

Contents

1	Introduction 1.1 Assumptions and Justifications	5 5
2	Task 1: Team Selection 2.1 Variable Definitions 2.2 Creating a Formula 2.3 Considering Restrictions 2.4 Defining Scaling Factor λ 2.5 Implementing Formula 2.6 Team Selection	6 6 7 7 9
3	Task 2: Tournament Schedule3.1Relaxed Mirrored Double Round Robin Structure3.2Considerations in Schedule Making3.3Approach 1: Berger Tables Method3.3.1Evaluation of Approach 13.2.2Applications of Approach 13.3.2Appiroach 2: Selective Annealing Programming3.4.1Defining Variables3.4.2Setting Constraints3.4.3Applying Weights to Factors Through Analytical Hierarchy Process (AHP)3.4.4Objective Function Calculation3.4.5Explanation of Simulated Annealing Programming (See Appendix C)3.4.6Normalising Factors3.4.7Final Result3.4.8Sensitivity Analysis of Objective Function3.5Selecting Days of Play3.5.1Approach 1: Same Local Time Schedule3.5.3Final Selection of Days3.5.4Rescheduling3.6Evaluation of Model	 9 9 10 11 12 12 12 13 15 16 17 17 18 18 19 20 21
4	Task 3: Expanding the Model 4.1 Selecting Teams 4.2 Selecting Matches 4.3 Selecting Days of Play	21 21 22 22
5	Task 4: Generalising the Model5.1 Applications of Current Model5.2 Sports With Limited Global Reach5.3 Sports Played Over Multiple Days5.4 Racing Sports	 23 23 24 24
6	References List	25
7	AI Use Report	26
8	Appendix 8.1 Appendix A: Data for Team Selection 8.2 Appendix B: Random Index Values 8.3 Appendix C: Selective Annealing Program Code 8.4 Appendix D: Carry Over Effects Matrix Generator	36 36 37 38 46

1 Introduction

For the Global Sports League's sport of choice, we have selected **association football**, also known as soccer and football. Throughout this paper, we will refer to the sport as **football**. The Global Sports League will consist of two stages: the **Round Robin stage** and the **Finals stage**. The first stage will be a relaxed mirrored double round robin where each team will play every other team twice (once at home, and once at away). Additionally, for calculation purposes, there are two 'ghost' teams that do not actually exist: if a team is meant to play against these ghost teams, they receive a bye and do not have to play against a team in that round. In total, each real team will receive four byes throughout the entire league. As such, the number of games played is ${}^{22}P_2 = 462$ games in total. As each game involves two teams, each team will play 462/(22/2) = 462/11 = 42 games. Another calculation that verifies this is $21 \times 2 = 42$ (each team will play the other 21 teams, including the two ghost teams, twice). Therefore, the Round Robin stage will include 42 rounds, during which each team will play a game with another team for a total of 11 concurrent games. The teams will be given points depending on the result of their games, as detailed in the table below:

Outcome	Points Given
Win	+3
Draw	+1
Loss	0

Table 1: Number of Points allocated per Game Result

From these points, the top four teams will advance to the Finals stage. The finals stage will comprise of two rounds - the Semifinals and the Finals. The 1st seed of the Round Robin stage will be matched with the 4th seed, and so on and so forth with the 2nd and 3rd seeds in the Semifinals. The winners of each semi-final will advance to the Finals, and the winner of the Finals will appropriately be crowned Champion of the Global Sports League. All of these games will be best of 1.

1.1 Assumptions and Justifications

1. **Stadium Use**: We assume that a home game will be hosted in the largest football stadium in that team's country.

Justification: This assumption is reasonable as (1) the national football team is generally associated with the largest stadium in a country, and (2) this maximises ticket sales and thus profit, benefiting the economic sustainability of the GSL.

2. Consistency of Travel Times: We assume that travel is directly proportional to distance between stadiums. This means that the model requires an input of team home stadium locations only.

Justification: (1) Since this is an international tournament consisting of national teams, it is reasonable to assume that the primary method of travel will be by plane, and travel speeds will be roughly the same. (2) It is reasonable to assume the national stadium of teams are located in the country's largest city, which is a convenient distance to the airport. (3) It is reasonable to assume travel between airports is streamlined, due to the status and resources of the national team. (4) This simplifies inputs required by the model, by avoiding unpredictable coach and private travel times.

3. Use of non-game days: We assume teams will be able to use all non-game days for rest or training, which will be beneficial for their next game.

Justification: This allows the simplification of assigning reasonable breaks between games, by eliminating the consideration of uncontrollable and unpredictable scheduling of travel, or other activities which are decided on a team by team basis.

2 Task 1: Team Selection

This league will feature teams from around the world, emphasizing geographic diversity, competitive equity, and logistical feasibility. The league should also ensure fair match ups, and include 20 teams with at least two teams from each continent (excluding Antarctica).

2.1 Variable Definitions

Variable	Definition
$N_{continent}$	Number of teams assigned to continent
N_{total}	Total number of teams in GSL
N_{top}	Number of teams in top 48 from continent
$P_{continent}$	Population of continent
P_{world}	Population of world
$FQF_{continent}$	FQF score for continent

 Table 2: Variable Definitions for Team Selection

2.2 Creating a Formula

To achieve a balance between number of teams from a continent, and total population of a continent, the following formula should be fulfilled:

$$\frac{N_{continent}}{N_{total}} = \frac{P_{continent}}{P_{world}}$$

$$N_{continent} = 20 \times \frac{P_{continent}}{P_{world}}$$
(1.1)

We must also prevent an over-representation of low quality teams. In order to do this, we introduce a Football Quality Factor (FQF), which accounts for the FIFA rankings of teams within a continent. FQF represents the proportion of the top 48 teams (in terms of FIFA points) which are from a continent.

Justification: Generally, continents with larger populations have more countries (Asia is the exception). If we consider the proportion of all FIFA teams (in terms of FIFA points) which are from a continent, this will favour continents with more FIFA teams, and thus a larger population. The top 48 teams are considered, as this is the cutoff for participation in the 2026 FIFA World Cup.

$$FQF_{continent} = \frac{N_{top}}{48} \tag{1.2}$$

The quality of a team is essential in ensuring competitiveness and fairness across the GSL, and higher quality teams will generate more engagement with the GSL. Hence, we prioritise quality of a team over continent population, with a weighting of 2-1.

$$N_{teams} = 20 \times \left(\frac{1}{3} \times \frac{P_{continent}}{P_{world}} + \frac{2}{3} \times FQF_{continent}\right)$$
(1.3)

#1000110649

2.3 Considering Restrictions

However, if we are to ensure at least 2 teams from every continent plays, the formula above results in the total number of teams exceeding 20. We must then introduce a scaling factor, applying only to teams assigned $N_{raw} < 2$.

Let us redefine the formula above:

$$N_i = 20 \times \left(\frac{1}{3} \times \frac{P_i}{P_{world}} + \frac{2}{3} \times FQF_i\right)$$

 N_i is number of teams allocated to a continent, without the min. 2 constraint i indexes the continents

The selective application of the scaling factor can be represented by a piecewise function:

$$N_{final} = \begin{cases} 2 & N_i < 2\\ \lambda & N_i & N_i \ge 2 \end{cases}$$

2.4 Defining Scaling Factor λ

Let C_{low} be the set of continents where $N_i < 2$. By Formula 3, we have allocated $2 \times |C_{low}|$ teams to C_{low} . Thus, the number of teams we have left to distribute is:

$$T_{left} = 20 - 2 \times |C_{low}| \tag{1.4}$$

Let C_{high} be the set of continents where $N_i \ge 2$. Let the uncapped total number of teams allocated to C_{high} by T_{high} . This is:

$$T_{high} = \sum_{i \in C_{high}} N_i \tag{1.5}$$

The scaling factor should adjust uncapped total number of teams to fit within the number of teams left to distribute.

$$\lambda \times T_{high} = T_{left}$$

$$\lambda = \frac{T_{left}}{T_{high}}$$
(1.6)

$$\sum_{i \in C_{low}} 2 + \sum_{i \in C_{high}} \lambda \times N_i = 20$$
(1.7)

2.5 Implementing Formula

Numbers rounded to 2 decimal points for simplicity. This is as the final number is rounded to the nearest whole number in the end, and hence additional accuracy is not necessary. See Appendix A for data used.

$$N_{Asia} = 20 \times \left(\frac{1}{3} \times \frac{4800}{8034} + \frac{2}{3} \times \frac{4}{48}\right) = 5.09$$
$$N_{Africa} = 20 \times \left(\frac{1}{3} \times \frac{1400}{8034} + \frac{2}{3} \times \frac{6}{48}\right) = 2.83$$

$$N_{Europe} = 20 \times \left(\frac{1}{3} \times \frac{750}{8034} + \frac{2}{3} \times \frac{26}{48}\right) = 7.84$$

$$N_{North\ America} = 20 \times \left(\frac{1}{3} \times \frac{600}{8034} + \frac{2}{3} \times \frac{4}{48}\right) = 1.61$$

$$N_{South\ America} = 20 \times \left(\frac{1}{3} \times \frac{440}{8034} + \frac{2}{3} \times \frac{7}{48}\right) = 2.31$$

$$N_{Oceania} = 20 \times \left(\frac{1}{3} \times \frac{44}{8034} + \frac{2}{3} \times \frac{1}{48}\right) = 0.31$$

The set C_{low} includes North America and Oceania. $|C_{low}| = 2$. Thus, we have allocated 4 teams to C_{low} .

$$T_{left} = 20 - 4 = 16$$
$$T_{high} = 5.09 + 2.83 + 7.84 + 2.31 = 18.07$$
$$\lambda = \frac{16}{18.07}$$
$$N_i \ge 2, \ N_{final} = \frac{8}{9} \times N_i$$

Continent	N_{final}
Asia	4.51
Africa	2.51
Europe	6.94
North America	2
South America	2.04
Oceania	2

Table 3: Teams per Continent After Scaling Factor Applied

The values of N_{final} are rounded to the nearest whole number, to find the final number of teams allocated to every continent.

Continent	No. Teams
Asia	5
Africa	3
Europe	7
North America	2
South America	2
Oceania	2

Table 4: Final Number of Teams Per Continent

2.6 Team Selection

Using FIFA rankings, the best national teams for each continent are chosen. We choose from best to worst, until the number of teams allocated to that continent has been chosen.

Asia	Africa	Europe	N.A.	S.A.	Oceania
Japan	Morocco	France	United States	Argentina	Australia
Iran	Senegal	Spain	Mexico	Brazil	New Zealand
Korea	Egypt	England			
Qatar		Portugal			
		Netherlands			
		Belgium			
		Italy			

Table 5: 20 Team Selection



Figure 1: Distribution of 20 Selected Teams

3 Task 2: Tournament Schedule

3.1 Relaxed Mirrored Double Round Robin Structure

Our tournament schedule is based on a relaxed mirrored double round robin (M2RR) structure. The relaxed M2RR structure consists of two single round robins (SRR), where the second SRR is a mirrored version of the first SRR. This means that home and away locations are swapped for every match in the first SRR. A relaxed structure (opposite of compact structure) means that teams are given by sfor certain rounds. This structure was chosen because of the following:

- 1. By design, this structure ensures each team or participant has an equal number of home and away games over the GSL.
- 2. Each team plays all other teams twice, in home and away venues. This benefits competitive equity, as no team will gain an advantage from not playing a strong team, or playing a team only at their home venue.

3. The provision of byes with a relaxed M2RR improves player performance by allowing more rest time between games, which leads to higher-quality performances.

3.2 Considerations in Schedule Making

In our model, we consider the following factors when determining matches and allocating home/away games in our relaxed M2RR schedule.

1. **Distance:** The total distance travelled by all teams through the GSL. Ideally, total distance travelled is as low as possible.

Explanation: Shorter travel distances help reduce travel-related stress and fatigue, improving overall performance of all teams. This also benefits the environmental friendliness of the GSL by minimising emissions from air travel, and increases logistical feasibility.

2. Breaks: A break occurs when a team has 2 consecutive home or away games. Consecutive breaks occur when a team has 3 or more consecutive home or away games. Ideally, teams alternate between home and away games and experience no breaks.

Explanation: This minimises continuous home-field advantage, which unfairly boosts performance due to familiarity with the venue, number of supporters, and reduced travel fatigue. Additionally, it promotes ticket sales, as people are less likely to watch consecutive matches with the same team at their city's stadium.

3. Carry Over Effect: The total carry over effect experienced by all teams through the GSL. Ideally, this is minimised.

Explanation: If a team plays against a weak opponent, it is likely to have a better outcome in the next round, than if it had played against a hard opponent. This is as teams who played a strong opponent in the previous round are likely to be more tired, and have lower morale.

4. **Disparity:** The disparity of all the above factors between all teams. Ideally, this is not present, to ensure competitive equity.

3.3 Approach 1: Berger Tables Method

The matches of a time slot correspond to the pairs of vertices that are adjacent to each other. A game always takes place at the venue of the team assigned to the head of the corresponding arc. The first 3 matches are illustrated in Figure 1. The schedules for the other time slots are derived analogously. The only difference is that the orientation of the arc incident to a select team changes each time. We use the Berger Tables method to create a schedule for 22 teams; 20 playing teams and 2 ghost teams that represent byes.



Figure 2: Example Utilisation of Berger Tables with 20 Teams

Once the first 21 rounds are allocated, these matches are mirrored in the last 21 rounds, with the allocation of home and away games switched around for every match.

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22	R23	R24	R25	R26	R27	R28	R29	R30	R31	R32	R33	R34	R35	R36	R37	R38	R39	R40	R41	R42
1	22	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	22	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20
2	21	22	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	22	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19
3	20	1	22	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	22	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18
4	19	21	2	22	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	22	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17
5	18	20	1	3	22	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	22	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16
6	17	19	21	2	4	22	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	22	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15
7	16	18	20	1	3	5	22	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	22	9	11	13	15	17	19	21	2	4	6	8	10	12	14
8	15	17	19	21	2	4	6	22	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	22	10	12	14	16	18	20	1	3	5	7	9	11	13
9	14	16	18	20	1	3	5	7	22	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	22	11	13	15	17	19	21	2	4	6	8	10	12
10	13	15	17	19	21	2	4	6	8	22	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	22	12	14	16	18	20	1	3	5	7	9	11
11	12	14	16	18	20	1	3	5	7	9	22	13	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	22	13	15	17	19	21	2	4	6	8	10
12	11	13	15	17	19	21	2	4	6	8	10	22	14	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	22	14	16	18	20	1	3	5	7	9
13	10	12	14	16	18	20	1	3	5	7	9	11	22	15	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	22	15	17	19	21	2	4	6	8
14	9	11	13	15	17	19	21	2	4	6	8	10	12	22	16	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	22	16	18	20	1	3	5	7
15	8	10	12	14	16	18	20	1	3	5	7	9	11	13	22	17	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	22	17	19	21	2	4	6
16	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	22	18	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	22	18	20	1	3	5
17	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	22	19	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	22	19	21	2	4
18	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	22	20	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	22	20	1	3
19	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	22	21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	22	21	2
20	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	22	1	3	5	7	9	11	13	15	17	19	21	2	4	6	8	10	12	14	16	18	22	1
21	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	22	2	4	6	8	10	12	14	16	18	20	1	3	5	7	9	11	13	15	17	19	22
22	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21

Figure 3: Sample Schedule with Teams 1-20, with 2 Ghost Teams in White

3.3.1 Evaluation of Approach 1

• **Minimising Breaks:** The Berger Tables approach provides a schedule with the smallest possible number of breaks.

The minimum total number of breaks for a M2RR with an even number of teams is 3n - 6, where n is the number of teams [17]. This number of breaks is provided with the Berger Tables approach. This is as teams with 1 break in the first half have a corresponding break in the second half, and a third break at the beginning of the second half. The first half must have 1 break to allow a team to play all teams (having a perfect AHAHA... pattern, where A denotes away games and H denotes home games, for every team limits each team to playing every other second team).

• **High COE:** The Berger Tables approach produces large COEs for each team. This is because there are multiple long sequences of teams facing opponents in nearly identical cyclical orders, which leads to repeated carry-over effects between many pairs of teams.

• **Distance Disregarded:** In the Berger Tables approach, the near-perfect AHAHA... pattern also maximises the distance travelled, as each team must travel back to their home stadium beteween each away game, instead of grouping away games that are a short distance from each other together.

3.3.2 Applications of Approach 1

We use the schedule generated by the Berger Method as our first base schedule in the selective annealing program, to which perturbations are applied (see Section 5.4).

3.4 Approach 2: Selective Annealing Programming

3.4.1 Defining Variables

Symbol	Definition
R	No. of rounds in GSL
m	No. of matches per round
g_{minH}	Min. number of home games for a team
g_{maxH}	Max. number of home games for a team
g_{CH}	No. consecutive home games for a team
g_{CA}	No. consecutive away games for a team
b _{total}	Total number of breaks in GSL
$\hat{b_T}$	No. breaks for a team
b_D	Disparity of breaks between teams
d_{total}	Total travel in GSL
$\hat{d_T}$	Total travel of a team
d_D	Disparity of travel times between teams
c _{total}	Total carry over effect in GSL
$\hat{c_T}$	Carry over effect for a team
c_D	Disparity in carry over effect between teams

Table 6: Variables for Selective Annealing Programming

3.4.2 Setting Constraints

A series of hard and soft constraints are used to guide the simulated annealing towards the optimal solution. If an iteration violates a hard constraint, it is considered invalid. A soft constraint is a preferential requirement, and not necessary. If an iteration violates a soft constraint, it is penalised.

A schedule is feasible if: (1) In each round, each team is in one match, and (2) after all rounds, each pair of teams has been in a match

C_{01}	R = 38
C_{02}	m = 10
C_{03}	$g_{minH} = 19$
C_{04}	$g_{maxH} = 19$

 Table 7: Hard Constraints

C_{05}	Minimise $\hat{b_T}$
C_{06}	Minimise b_D
C_{07}	Minimise \hat{d}_T
C_{08}	Minimise d_D
C_{09}	Minimise $\hat{c_T}$
C_{10}	Minimise c_D

 Table 8: Soft Constraints

3.4.3 Applying Weights to Factors Through Analytical Hierarchy Process (AHP)

We develop a 3×3 pairwise comparison matrix M to assign relative importance to the three factors affecting how optimal a schedule is. The matrix contains elements m_{ij} , each indicating the factor *i* relative to *j*. We will use a scale of 15 for the values m_{ij} , with 1 indicating equal importance and 5 indicating extreme disparity in importance. For simplicity in notation, single letters will be assigned to each factor:

D	Distance
В	Breaks
C	Carry Over Effect

Table 9: Simplified Symbol Allocation for Comparative Matrix

- 1. $m_{DB} = \frac{1}{3}$ Justification: Breaks are more significant than distance. Equality of breaks is essential to competitive equity, which is necessary to produce a valid tournament winner. Additionally, minimising breaks benefits ticket sales and thus economic sustainability of the tournament. On the other hand, disparity in travel distances does not significantly affect competitive equity. This is as although longer travel times may result in more fatigue for a team, we ensure at least 5 days between each match (see Section 3.5). Thus, teams have a generous amount of time to recover from any travel fatigue, and any disparity in initial travel distance will have little affect in a team's performance in the next match.
- 2. $m_{DC} = \frac{1}{4}$ Justification: COE is much more significant than distance. This is as COE is slightly more significant than breaks (see below) and breaks are more significant than distance (see above).
- 3. $m_{BC} = \frac{1}{2}$ Justification: Conventionally, the minimisation of breaks is prioritised over the minimisation of COE. This is as in regular international tournaments, the quality of teams are usually all very high (the host nation's team may be an exception), and COE is minimal as teams are often equally matched. However, due to the mandatory 2 teams from every continent, countries with significantly lower FIFA rankings (eg. Uzbekistan and New Zealand) are included in the GSL, and this large disparity in skill level increases the affect of COE significantly.

Thus, our comparative matrix is as follows:

$$M = \begin{cases} 1 & m_{DB} & m_{DC} \\ \frac{1}{m_{DB}} & 1 & m_{BC} \\ \frac{1}{m_{DC}} & \frac{1}{m_{BC}} & 1 \end{cases} = \begin{cases} 1 & \frac{1}{3} & \frac{1}{4} \\ 3 & 1 & \frac{1}{2} \\ 4 & 2 & 1 \end{cases}$$

To find the eigenvalues of the matrix M, we use the equation;

$$det(M - \lambda I) = 0$$

where: I = an identity matrix the same size as M (ie. 3×3) det = determinant

The 3 \times 3 matrix results in a 3rd degree polynomial $\lambda^3 - 3\lambda^2 - \frac{1}{6}$, which is solved numerically.

$$\lambda_1 \approx 3.018, \ \lambda_2 \approx -0.009 + 0.233i, \ \lambda_3 \approx -0.009 - 0.233i$$

$$\lambda_{max} = 3.018$$

For each eigenvalue λ we solve the corresponding eigenvector v using the equation;

$$(M - \lambda I)v = 0$$

The largest eigenvalue λ_{max} corresponds to the principal eigenvector. The eigenvector associated with λ_{max} is then normalised (so its components sum to 1);

$$w_i = \frac{v_i}{\sum v_i}$$

$$w = [0.122, 0.320, 0.558]$$

In the context of the objective function (see Section 3.4.4), $\alpha = 0.122, \beta = 0.320, \gamma = 0.558$.



To check the judgement matrix's consistency, we will use the formulae;

$$CR = \frac{CI}{RI} \tag{3.1}$$

where: CR = Consistency RatioCI = Consistency IndexRI = Random Consistency Index

For a $n \times n$ matrix:

$$CI = \frac{\lambda_{max} - n}{n - 1}$$
(3.2)
$$CI = \frac{3.018 - 3}{3 - 1} = 0.009$$

RI is the average CI for a large sample of randomly generated matrices of size n. The retrieved value of RI for n = 3 is $RI \approx 0.58$ (See Appendix B).

$$CR = \frac{0.009}{0.58} \approx 0.0155$$

 $CR \leq 10\%$ indicates acceptable consistency in the comparative matrix [2]. Thus, our comparative matrix can be considered very consistent.

3.4.4 Objective Function Calculation

Score to be optimised by the objective function:

$$Penalty = \alpha(\sum_{t} \hat{d}_{T} \cdot d_{D}) + \beta(\sum_{t} \hat{b}_{T} \cdot b_{D}) + \gamma(\sum_{t} \hat{c}_{T} \cdot c_{D})$$
(2.1)

where: $\alpha, \beta, \gamma = \text{scaling constants}$

$$d = 2r \cdot \arcsin(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2} + \cos(\phi_1) \cdot \cos(\phi_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})})$$
(2.2)

where: d_{ij} = distance from team i to team j

 ϕ_1 and $\phi_2 =$ latitudes of point 1 and point 2 in radian form

 λ_1 and $\lambda_2 =$ longitudes of point 1 and point 2 in radian form

$$c_j = \sum_i w_{ij} \cdot (c_{ij})^2 \tag{2.3}$$

where: $c_j = \text{total COE experienced by team } j$

 $w_{ij} =$ strength of team *i* relative to team *j* (See Appendix D)

 c_{ij} = number of COEs experienced by team *j* from team *i*

The weights of team strength were calculated by accounting for match statistics across previous FIFA World Cup tournaments (or estimations of the statistics based on international performances as shown in the rightmost column). The statistics factored in **Scoring Efficiency** (*S.E.*), **Match Control** (*M.C.*), **Game Engagement** (*G.E.*), **Attacking Pressure** (*A.P.*) and **Possession Management** (*P.M.*). These factors were split into positively and negatively impacting metrics (e.g. *S.E.* split into Average Goals per Match and Average Goals Against (conceded) per Match) and formed into radar graphs. Finally, the strength of each team was calculated by normalising the difference in areas between the positively and negatively impacting graphs. Therefore, a weight w_{ij} could be expressed as $\hat{w}_i - \hat{w}_j$, where \hat{w}_i and \hat{w}_j are the normalised strengths as calculated above of teams *i* and *j* respectively.

		Avg. Goals/Match	Avg. Goals Against/Match	Avg. Points/Match	Yellow Cards	Games Played	Fouls Comitted	Shots On Goal	Offsides	Pass Completions	Possession Lost	Reference Competitions Used for Statistical Estimates
	Japan	2.51	0.62	2.17	4	120	47	13	8	1273	120	
	Iran	2.29	0.81	2.02	4	123	37	8	4	1028	98	
Anin	Korea	1.97	0.83	2.01	2	133	33	13	2	1295	110	
Asid	Qatar	2.06	1	1.77	6	97	45	8	5	1071	115	
	Iraq	~1.8	~0.9	1.71	~2.0	102	~14	~5.5	~2.4	~640	~115	2023 AFC Asian Cup, 2026 WCQ Matches
	Uzbekistan	~1.7	~0.8	1.81	~1.8	80	~12	~4.8	~2.3	~710	~110	2023 AFC Asian Cup, 2026 WCQ Matches
	Morocco	1.58	0.69	1.83	5	100	77	15	19	2077	130	
Africa	Senegal	1.71	0.95	1.77	8	65	61	13	7	1366	125	
	Egypt	~1.6	~0.7	1.73	~2.2	75	~15	~5.2	~1.9	~480	~105	2023 AFCON (Africa Cup of Nations), 2026 WCQ Matches
	France	1.97	1.09	1.92	5	119	63	31	16	2436	140	
	Spain	2.2	0.94	2	3	131	40	28	5	3041	85	
	England	1.99	0.75	1.97	1	137	45	24	7	2430	95	
	Portugal	2.15	0.79	2.09	6	106	65	22	8	2121	105	
Europe	Netherlands	2.27	0.9	1.98	11	115	55	17	8	2071	115	
	Belgium	1.95	1.06	1.78	2	120	35	11	1	1448	90	
	Italy	~1.5	~0.9	2.04	~2.5	136	~13	~5.5	~2.5	~550	~100	2024 UEFA Euro Qualifiers, 2022-23 UEFA Nations League
	Germany	2.46	1.12	2.17	3	156	29	22	5	1985	100	
	Croatia	1.91	0.83	2.07	9	77	81	16	6	2434	135	
NI A	United States	1.55	0.99	1.63	6	125	50	11	9	1720	110	
IN.A.	Mexico	1.92	1.1	1.75	6	144	49	11	10	1594	105	
	Argentina	1.78	1.06	1.93	10	189	88	49	10	3649	145	
3.A.	Brazil	2.15	0.85	2.14	5	201	57	27	9	2685	120	
Occorrio	Australia	2.72	1.02	2.01	5	113	50	9	6	1083	130	
Oceania	New Zealand	~2.0	~0.6	1.85	~17	55	~11	~4.6	~20	~470	~110	OEC Nations Cup. Intercontinetal Playoffs 2026 WCO

Figure 4: Team Strength Data for COE Weightings







Figure 6: Mexico National Team Strength Distribution



Figure 7: Portugal National Team Strength Distribution

3.4.5 Explanation of Simulated Annealing Programming (See Appendix C)

Simulated annealing follows a similar process to physical annealing of metal, where it is gradually cooled to form a crystalline structure. Here, random perturbations will be induced in our schedule, for this specific example, it will swap home and away allocations to each matchup, and also swap the order of the rounds. Swapping the home and away allocations is intended to reduce the distance and also the breaks, by randomising where each team has to travel, whereas swapping rounds as a whole is intended to reduce distance and the carry over effect, by randomising the order each team plays every other team.

Then, this new temporary schedule will be evaluated by the Penalty formula above, then compared with the old schedule. If the new schedule is better, it is accepted and this process starts again. However, if it is worse it will revert back to the original schedule most of the time, but it will still have a chance inversely proportional to the 'temperature' or the time this program has been running to accept the worse schedule. This is done to ensure that the program explores other branches of possible schedules so that it does not just converge at a local minima.

3.4.6 Normalising Factors

To normalise distance, breaks and COE, we use the generic min max normalisation formula:

$$\hat{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Since we are creating a new schedule in every iteration of the selective annealing programming, the min and max of each factor are unknown. To overcome this, we update the minimum and maximum of each factor after every iteration and calculation, such that the normalisation is dynamic. This will have little effect on the validity of our Penalty score.

3.4.7 Final Result



Figure 8: Penalty Score over 50,000 Iterations

Here, the graph of Penalty score over iterations spikes up drastically before trending downwards. This is because, according to our model, our initial schedule using the Berger Tables Method already has a significantly low penalty. Our simulation has to first escape the local minima and explore other, perhaps suboptimal, solutions, before gravitating towards the global minima. The resulting matches are illustrated in our visual graphic.

3.4.8 Sensitivity Analysis of Objective Function

The most significant uncertainty factor in our objective function is the weightings of each of the three factors, as although justified, in reality these values are determined by the GSL administrative body. To analyse the sensitivity of our objective function, we employ the one-at-a-time variation local sensitivity analysis method. We then perturb each weight factor α, β, γ by $\pm 20\%$, and observe changes to the penalty value (the base schedule is inputted).

Case	α	β	γ	Penalty
Baseline	0.122	0.320	0.558	11.39
$\alpha - 20\%$	0.098	0.329	0.573	12.54
$\alpha + 20\%$	0.146	0.311	0.543	11.20
$\beta-20\%$	0.129	0.256	0.615	9.80
$\beta + 20\%$	0.108	0.384	0.508	12.02
$\gamma-20\%$	0.138	0.361	0.501	11.91
$\gamma + 20\%$	0.100	0.276	0.624	10.85

Table 10: Perturbations for One-at-a-time Sensitivity Analysis

We can see that under these significant perturbations, the penalty values never change significantly. This demonstrates the reliability and stability of our objective function used to direct the simulated annealing program.

3.5 Selecting Days of Play

The league consists of 20 teams, and each team must play every other team twice as this is a mirrored double round robin tournament. As discussed above, there are 42 rounds of 11 games in the Round Robin stage.

The allocated time is 8-9 months: in order to maximise flexibility, for the 20-team league, the tournament length will be set to 8 months. It has been calculated that 8 months will approximately correspond to 243 days, as $365 \ge 8/12 = 243.333...$ As such, one game should be played every 243/42 = 5.79 days (2d.p.). Keeping this in mind, the necessary criteria for a good schedule is as follows:

- 1. The schedule should not exceed 243 days. It should also leave sufficient time to account for misadventures and vacations for the players.
- 2. The schedule should include 42 games with a relatively consistent frequency.
- 3. There should be a sufficient break between games to ensure proper recuperation. We have determined that a buffer of at least **5 days** is necessary to ensure the wellbeing of players, especially considering the strenuous requirements of global travel. Additionally, the **rest difference** between teams should be minimal. The rest difference is the difference between the amount of rest time each time received.
- 4. The schedule should prioritise the days of highest stadium attendance and viewership to maximise profits. The weekend (Saturday and Sunday) attracts the highest number of fans, followed by Tuesday, Wednesday, Monday, Thursday and Friday.

With these criteria in mind, there are two possible approaches for formulating the days on which games are played.

3.5.1 Approach 1: Same Local Time Schedule

In this approach, all of the games in a 'round' (11 games in total) would be played on the same day and time for **each country's local timezone**. What this entails is that, for example, if a game was played at 5pm on Saturday in the USA, a game would be played on the same Saturday at 5pm in China, but by China's local time. Through this method, all of the games will be held at the same time and day, reducing the variability of football games and making it more equitable. Furthermore, with this method,

there would be significantly less rest difference, as the rest difference would be determined only by the difference in time zones. As such, rest difference is at maximum 13 hours, which is minor compared to the minimum 120 hours allocated for rest time. Thus, this method is highly equitable. However, this method does come with the caveat of reducing viewing experience as a whole, for two main reasons.

- 1. Although this is not an issue **across** time zones, in a place such as Europe where there are a large concentration of teams in the same time zone, it is inevitable that two or more games will be happening at the same local and global time. As such, it would be impossible to watch all of the games of the GSL live without having to watch two games simultaneously. However, this is a small issue. Whilst domestic leagues such as the English Premier League somewhat stagger the days of play of their matches to minimise simultaneous viewership, they also suffer from this same issue, meaning that fans will be used to having to sacrifice watching a team live.
- 2. Another issue that arises from this schedule is that the games will be less equally distributed throughout the week, hence reducing the amount of football content available to fans in between games. However, once again, this is a trivial problem. Fans will have sufficient content from the 10 games played in each round to adequately bide them over until the next set of games.

Whilst both of this issues would theoretically be resolved by staggering the days of play, this ultimately causes more problems.

3.5.2 Approach 2: Staggering the Days of Play and Minimising Rest Mismatches

As discussed previously, staggering the days of play so that games are held throughout the week is able to improve the viewing experience of the GSL by better distributing football games to ensure a steady flow of content to fans. However, this approach comes with sacrifices to equity that leads to this scheduling method being inferior to Option 1.

- 1. Unlike Option 1, which preserves the same local time and day to attempt to create equity in the stadium and player conditions for every team, this method requires inconsistencies in playing day and time. It has been found that playing at different times of day impacts performance: games held near the time of day where the circadian peak occurs had teams performing better than anticipated. This introduces an unpredictable element into the schedule that reduces equity. In our model, we prioritise equity over viewing experience.
- 2. Option 1's rest difference is largely equitable as the maximum rest difference is approximately 10 percent of the minimum rest time. However, with this schedule, a layer of complexity is added where this is not necessarily guaranteed. In a domestic league, this issue does not exist as the amount of travel and time zone difference you have is limited. However, in the Global Sports League, there is a significant amount of time devoted to travelling and time zone that is highly variable and changes per match. Therefore, if the days are staggered, there are a limited number of ways to ensure that each team has a minimum of 5 days rest between games. To allow for this and minimise disparities in breaks and carry-over effect is heuristically impossible with this approach and not reasonable to implement.

Therefore, whilst Approach 2 will theoretically improve the experience for the fans, we have prioritised equity over viewing experience. Ensuring that every team is subject to equal conditions will, in our opinion, be the most crucial factor in creating a good league and keeping fans satisfied. Thus, we have opted to use **Approach 1** of keeping all teams playing in the same time and day for their local region to maximise equality.

3.5.3 Final Selection of Days

With this conclusion, we began to consider the cycle on which the Global Sports League. As discussed above, Saturday is the most popular day for football games. Thus, the Global Sports League will begin on a Saturday to garner maximum interest on the opening games. Having a good start is critical to ensure a sports league is popular and optimise profits. Rest periods should be either 5 or 6 days to leave sufficient days to account for misadventure and holidays. From Saturday, this can result in the next game being played on a Thursday or Friday. As Thursday has greater viewership, it is prioritised. From Thursday, the next game can be scheduled on a Tuesday or Wednesday. Tuesday is prioritised. From Sunday, the next game can be scheduled on either Sunday or Monday. Sunday is scheduled next. From Sunday, the next game is on either Friday or Saturday. Saturday is the next day in the cycle, returning to the beginning of the loop. Visually depicted, it appears as follows:



Figure 9: Day Selection Cycle

With this cycle, 4 rounds will be played within 21 days, with an average of one round every 5.25 days, thus being able to fit within 8 months with plenty of leeway for unforeseen circumstances. Within 210 days, 10 cycles of the schedule will have been completed, completing 40 rounds and leaving 2 games left in the league. At this point, it would be a Saturday on the day of the 41st round. As such, the league will conclude on a Thursday within 215 days of the league's beginning, assuming that there has been no incidents causing rescheduling. Of the allocated 8 months, four weeks/28 days remain. This leaves two weeks/14 days to complete the semi-finals and the finals, and two weeks/14 days of misadventure/vacation.

3.5.4 Rescheduling

In the event of misadventure, the response varies depending on how far the misadventure sets a match back.

- 1. If the match is delayed by three or more days, the match is cancelled entirely. The loss will be given to the team who caused the delay, if applicable (e.g. if a team was unable to enter a country due to not filling out the appropriate paperwork). If it was not one team's fault necessarily (e.g. rain on the game day) the game would be ruled a draw.
- 2. If the match is delayed by two days or less, the match will be postponed to the next possible time. Whilst this would cause slight inequities in rest time, this can be remedied through a number of countermeasures such as:
 - Interjecting week-long break periods when multiple instances of postponing occur, up to the tournament organisers' discretion
 - Recognising that the team will have a longer rest period BEFORE they play their postponed match, meaning that the rest difference roughly cancels out in aggregate

3.6 Evaluation of Model

Advantages:

- Dynamic probabilistic acceptance of worse solutions dependent on the 'temperature' of the system allows it to escape local minima for exploration of a broader solution space, converging towards optimal minima
- Simplicity in logic and implementation
- Robust performance, although lengthy, does not have significant deviation for larger scale optimisation (e.g. 20 to 24 teams)
- Low sensitivity of Penalty function means parameters and scaling factors do not have to be finetuned for meaningful optimisation

Disadvantages:

- Lengthy computing time to achieve accurate and optimised results
- Slower convergence rate than more deterministic methods like gradient-based optimisation

4 Task 3: Expanding the Model

4.1 Selecting Teams

$$N_{Asia} = 24 \times \left(\frac{1}{3} \times \frac{4800}{8034} + \frac{2}{3} \times \frac{4}{48}\right) = 6.11$$

$$N_{Africa} = 24 \times \left(\frac{1}{3} \times \frac{1400}{8034} + \frac{2}{3} \times \frac{6}{48}\right) = 3.39$$

$$N_{Europe} = 24 \times \left(\frac{1}{3} \times \frac{750}{8034} + \frac{2}{3} \times \frac{26}{48}\right) = 9.41$$

$$N_{North\ America} = 24 \times \left(\frac{1}{3} \times \frac{600}{8034} + \frac{2}{3} \times \frac{4}{48}\right) = 1.93$$

$$N_{South\ America} = 24 \times \left(\frac{1}{3} \times \frac{440}{8034} + \frac{2}{3} \times \frac{7}{48}\right) = 2.77$$

$$N_{Oceania} = 24 \times \left(\frac{1}{3} \times \frac{44}{8034} + \frac{2}{3} \times \frac{1}{48}\right) = 0.38$$

The set C_{low} includes North America and Oceania. $|C_{low}| = 2$. Thus, we have allocated 4 teams to C_{low} .

$$T_{left} = 24 - 4 = 20$$

$$T_{high} = 6.11 + 3.39 + 9.41 + 2.77 = 21.68$$

$$\lambda = \frac{20}{21.68}$$
for C_{high} , $N_{final} = \frac{20}{21.68} \times N_i$

Continent	N_{final}
Asia	5.64
Africa	3.13
Europe	8.68
North America	2
South America	2.56
Oceania	2

Table 11: Teams per Continent After Scaling Factor Applied

The values of N_{final} are rounded to the nearest whole number, to find the final number of teams allocated to every continent. Note that South America's 2.56 has been rounded down to maintain 24 total teams.

Continent	No. Teams
Asia	6
Africa	3
Europe	9
North America	2
South America	2
Oceania	2

Table 12: Final Number of Teams Per Continent

Using FIFA rankings, the best national teams for each continent are chosen. We choose from best to worst, until the number of teams allocated to that continent has been chosen.

Asia	Africa	Europe	N.A.	S.A.	Oceania
Iraq		Germany			
Uzbekistan		Croatia			

Table 13: Additional teams from each continent

4.2 Selecting Matches

For a double mirrored round robin with 24 teams (26 teams: 24 total teams + 2 ghost teams which will count as byes), the initial schedule would need to be rewritten with the Berger Tables Method with 26 teams, with teams 25 and 26 representing byes. The teams variable would need to changed to 26, with the number of rounds being 50 rounds. Each function will also need to be updated to disregard teams 25 and 26 when calculating distance, breaks and carry over effects. The distance matrix will also need to be updated to include the 4 extra teams using the haversine formula. This will not be an arduous task due to the ubiquity of the information. Furthermore, the carry over weighting matrix will also need to be updated.

4.3 Selecting Days of Play

With an additional four teams playing in the tournament, the previous schedule must be updated. With 24 real teams and 2 ghost teams to represent byes, the number of games that each team has to play is $(26-1) \ge 50$. As such, we will have 50 rounds of 13 games, leading to a total number of 650 games. Each team will receive 4 byes, similarly to the initial model. The initial day cycle of Saturday - Thursday - Tuesday - Sunday - Saturday can be preserved: it would be impossible to reduce it to be

more efficient whilst still allowing sufficient rest for players. However, the excess month that was not used in the initial model must be applied here. As the cycle takes 21 days to complete 4 rounds, within 12 cycles or 252 days, 48 rounds will have been completed, with the league being on the day of the 49th round. Thus, the league will conclude on a Thursday, 257 days after the first round. It has been approximated that 9 months consist of 273 days, to take the conservative estimate. Thus, this leaves 16 days for semifinals, finals, misadventure and vacations. Whilst a seemingly insufficient amount, this is bolstered by the four byes that each team has. Additionally, if necessary, the semifinals and finals can be scheduled to be closer together in time to create further leeway.

5 Task 4: Generalising the Model

5.1 Applications of Current Model

Our current team selection and scheduling method is applicable for most team sports. This is as we can apply our current model for all team sports that fulfil the following criteria:

- 1. Has an international ranking/points system of national teams
- 2. Provides an outcome of win, lose or draw for each team at the end of a match
- 3. Matches are completable within one day
- 4. Has 20 or more national teams participating in the sport competitively

Team sports that meet these criteria include: cricket, hockey (field, ice, roller), volleyball (regular, beach, Sepak Takraw), basketball, baseball, rugby, Australian rules football, netball, softball, lacrosse, and polo.

5.2 Sports With Limited Global Reach

We must generalise our model to account for team sports that do not currently have sufficient international presence for a governing body that maintains a national team ranking/points system. For these team sports, we can instead hold continental round robin selection tournaments to determine our final GSL teams. All countries in each continent will be able to participate in the continental selection tournament. The schedule of the round robin can be determined using our model, with the number of teams modified. The points gained by each team over selection tournaments are then scaled to the number of teams within the tournament. The following GSL will be modelled with our current model, using scaled points from the selection tournaments as a replacement for national team rankings/points determined by a governing body.

In continents with a large number of countries, we must ensure the selection tournaments are not excessively lengthy. To do this, we hold regional round robin tournaments can be held instead. For example, Africa holds regional tournaments for North Africa, East Africa, Central Africa, West Africa, and Southern Africa.

Additionally, some teams sports with limited global reach may have less than 20 teams, meaning they do not require a team selection process, as all teams are automatically entered into the GSL. For these sports, we adjust our model and create a schedule for the number of available teams.

5.3 Sports Played Over Multiple Days

Additionally, we must generalise our model to account for team sports with matches that occur for more than one day (notably test cricket). This affects the Day Selection section of our model. For test cricket, games are held over 5 days and for more than 6 hours daily. As such, it requires a longer rest period between games. We have determined that a rest period of approximately 9 days is necessary to ensure that players are able to recuperate properly and have sufficient time to train between games. This places the schedule on a fortnightly basis, which is ideal for maximising profits by holding the game on the most popular days of the week. Additionally, it makes the game day very predictable and easy to remember for fans, improving the viewing experience. Generally, test cricket begins on a Thursday. This guarantees that the third and fourth days, the climax, are held on the weekend, thus garnering the most stadium and TV attendance. Another change to the schedule is that to complete the GSL in 8-9 months, to have 20 teams participating is impossible without giving the players insufficient rest. As such, the number of teams will have to be reduced to 10 or less.

Something similar can be done for all other sports that have games over multiple days.

5.4 Racing Sports

Team racing sports (notably F1) provide a unique challenge to our existing model, as their unique structure provides a number of implications.

- 1. The GSL utilises a series of rounds in different tracks around the world, where each racecourse is used once throughout the GSL. This is in contrast to the round robin format used in our original model.
- 2. Tracks used are not determined by the nationality of the qualifying teams. Instead, factors such as licensing, historical significance, fan appeal, and the value of bids from each track.
- 3. All disparity factors considered in our original model are irrelevant, as all teams will be playing in the same rounds, at the same locations.
- 4. Carry over effect also does not need to be considered for the same reason.
- 5. There are no breaks, as each racecourse is used only one time.
- 6. Total travel distances of all teams can be optimised, by adjusting the order in which the racecourses that are played in throughout the GSL.

Thus, for team racing sports, we can simplify the cost function of our model, to only consider total travel time.

6 References List

- URL: https://www.researchgate.net/publication/228690594_A_Framework_for_ Scheduling_Professional_Sports_Leagues
- 2. URL: https://www.researchgate.net/figure/Random-Index-RI-Saaty-1980_tbl2_ 323905554
- 3. URL: https://www.sciencedirect.com/science/article/abs/pii/S0377221722007792
- 4. URL: https://www.ticketgum.com/blog/most-football-crazy-countries
- 5. URL: https://www.jstor.org/stable/20141544
- URL: https://www.researchgate.net/publication/220270875_The_Traveling_ Tournament_Problem_Description_and_Benchmarks
- 7. URL: https://inside.fifa.com/fifa-world-ranking/men
- 8. URL: https://digitalhub.fifa.com/m/445599b755be1b8a/original/ Draw-Procedures-for-the-UEFA-Preliminary-Draw-for-the-FIFA-World-Cup-26.pdf
- 9. URL: https://www.ticketgum.com/blog/most-football-crazy-countries
- 10. URL: https://www.austadiums.com/sport/football/events
- 11. URL: https://www.espn.com.au/football/fixtures/_/date/20250308
- 12. URL: https://math.stackexchange.com/questions/2083611/round-robin-tournament-but-no-tea
- 13. URL: https://wikipedia.nucleos.com/viewer/wikipedia_en_all/A/List_of_domestic_ football_league_broadcast_deals_by_country
- 14. URL:+mm https://www.tandfonline.com/doi/full/10.1080/16184742.2022.2067208# abstract
- 15. URL: https://repository.brynmawr.edu/cgi/viewcontent.cgi?article=1007&context= econ_pubs
- 16. URL: https://internationalsports.nfshost.com/index.php/sports/era/recent_ decades/sepak_takraw/58
- 17. URL: https://mat.tepper.cmu.edu/trick/survey.pdf
- 18. URL: https://doi.org/10.5665/sleep.3248

7 AI Use Report

1. OpenAI ChatGPT (Dec 5, 2024 version, ChatGPT-o1)

Query: Rank Country Points Continent 1 Argentina 1867.25 South America 2 France 1859.78 Europe 3 Spain 1853.27 Europe 4 England 1813.81 Europe 5 Brazil 1775.85 South America 6 Portugal 1756.12 Europe 7 Netherlands 1747.55 Europe 8 Belgium 1740.62 Europe 9 Italy 1731.51 Europe 10 Germany 1703.79 Europe 11 Uruguay 1695.91 South America 12 Colombia 1694.44 South America 13 Croatia 1691.59 Europe 14 Morocco 1688.18 Africa 15 Japan 1652.79 Asia 16 United States 1645.48 North America 17 Senegal 1637.25 Africa 18 IR Iran 1635.31 Asia 19 Mexico 1627.40 North America 20 Switzerland 1625.16 Europe 21 Denmark 1611.49 Europe 22 Austria 1589.84 Europe 23 Korea Republic 1585.45 Asia 24 Ecuador 1560.13 South America 25 Ukraine 1554.94 Europe 26 Australia 1544.15 Oceania 27 Sweden 1540.20 Europe 28 Türkiye 1537.24 Europe 29 Wales 1534.39 Europe 30 Hungary 1517.57 Europe 31 Canada 1515.96 North America 32 Serbia 1514.28 Europe 33 Egypt 1513.48 Africa 34 Russia 1512.32 Europe 35 Poland 1510.62 Europe 36 Panama 1504.39 North America 37 Algeria 1495.85 Africa 38 Romania 1494.20 Europe 39 Greece 1489.88 Europe 40 Peru 1488.42 South America 41 Slovakia 1486.19 Europe 42 Czechia 1484.88 Europe 43 Norway 1484.44 Europe 44 Nigeria 1482.23 Africa 45 Scotland 1480.24 Europe 46 Côte d'Ivoire 1478.47 Africa 47 Venezuela 1475.71 South America 48 Qatar 1474.60 Asia 49 Cameroon 1471.49 Africa 50 Chile 1469.44 South America 51 Mali 1466.10 Africa 52 Tunisia 1464.42 Africa 53 Paraguay 1458.26 South America 54 Costa Rica 1454.99 North America 55 Slovenia 1454.25 Europe 56 Iraq 1442.86 Asia 57 South Africa 1428.87 Africa 58 Uzbekistan 1424.93 Asia 59 Saudi Arabia 1406.56 Asia 60 Republic of Ireland 1400.22 Europe 61 Congo DR 1395.77 Africa 62 Jamaica 1390.85 North America 63 UAE 1385.57 Asia 64 Jordan 1375.91 Asia 65 Albania 1375.57 Europe 66 Burkina Faso 1373.76 Africa 67 North Macedonia 1368.96 Europe 68 Georgia 1362.21 Europe 69 Finland 1361.53 Europe 70 Iceland 1355.10 Europe 71 Northern Ireland 1349.58 Europe 72 Cabo Verde 1328.51 Africa 73 Montenegro 1326.76 Europe 74 Bosnia-Herzegovina 1326.69 Europe 75 Honduras 1325.93 North America 76 Israel 1322.25 Europe (UEFA) 77 Ghana 1320.52 Africa 78 Guinea 1311.88 Africa 79 Bolivia 1308.58 South America 80 Oman 1306.67 Asia 81 Bahrain 1305.52 Asia 82 Bulgaria 1301.04 Europe 83 El Salvador 1300.81 North America 84 Gabon 1297.87 Africa 85 Angola 1296.48 Africa 86 Haiti 1287.75 North America 87 Zambia 1279.24 Africa 88 Uganda 1270.42 Africa 89 New Zealand 1269.83 Oceania 90 China PR 1266.38 Asia 91 Curação 1264.78 North America 92 Luxembourg 1256.42 Europe 93 Equatorial Guinea 1256.20 Africa 94 Benin 1250.99 Africa 95 Syria 1249.47 Asia 96 Mozambique 1232.70 Africa 97 Thailand 1231.17 Asia 98 Belarus 1226.31 Europe 99 Kosovo 1219.82 Europe 100 Armenia 1219.15 Europe 101 Palestine 1215.87 Asia 102 Trinidad and Tobago 1214.81 North America 103 Comoros 1208.99 Africa 104 Tajikistan 1203.08 Asia 105 Guatemala 1201.35 North America 106 Tanzania 1199.33 Africa 107 Kyrgyz Republic 1194.19 Asia 108 Kenya 1189.65 Africa 109 Mauritania 1183.16 Africa 110 Kazakhstan 1180.18 Europe 111 Namibia 1179.26 Africa 112 Lebanon 1171.43 Asia 113 Sudan 1165.41 Africa 114 Vietnam 1164.79 Asia 115 Korea DPR 1164.41 Asia 116 Madagascar 1162.02 Africa 117 Azerbaijan 1158.70 Europe 118 Libya 1158.03 Africa 119 Togo 1152.18 Africa 120 Guinea-Bissau 1151.72 Africa 121 Zimbabwe 1151.06 Africa 122 Niger 1145.96 Africa 123 Estonia 1142.53 Europe 124 Rwanda 1136.06 Africa 125 Gambia 1134.69 Africa 126 India 1133.62 Asia 127 Indonesia 1133.41 Asia 128 Congo 1132.65 Africa 129 Sierra Leone 1132.06 Africa 130 Cyprus 1131.32 Europe 131 Malawi 1129.31 Africa 132 Malaysia 1115.51 Asia 133 Nicaragua 1114.28 North America 134 Kuwait 1108.72 Asia 135 Central African Republic 1102.23 Africa 136 Botswana 1098.85 Africa 137 Faroe Islands 1096.88 Europe 138 Suriname 1096.60 South America 139 Burundi 1087.26 Africa 140 Latvia 1084.11 Europe 141 Dominican Republic 1083.36 North America 142 Lithuania 1069.96 Europe 143 Turkmenistan 1065.42 Asia 144 St. Kitts and Nevis 1064.99 North America 145 Liberia 1063.04 Africa 146 Ethiopia 1059.94

Africa 147 Solomon Islands 1051.00 Oceania 148 Fiji 1048.48 Oceania 149 Lesotho 1046.76 Africa 150 Philippines 1046.67 Asia 151 Moldova 1045.59 Europe 152 New Caledonia 1034.43 Oceania 153 Tahiti 1031.47 Oceania 154 Guyana 1030.60 South America 155 Hong Kong, China 1028.76 Asia 156 Afghanistan 1025.00 Asia 157 Puerto Rico 1022.83 North America 158 Yemen 1018.27 Asia 159 Eswatini 1015.25 Africa 160 Singapore 1011.68 Asia 161 Antigua and Barbuda 1010.97 North America 162 Maldives 1001.39 Asia 163 Cuba 994.43 North America 164 Vanuatu 994.22 Oceania 165 Chinese Taipei 990.56 Asia 166 Bermuda 988.19 North America 167 St. Lucia 985.72 North America 168 Malta 983.95 Europe 169 Myanmar 981.35 Asia 170 South Sudan 973.55 Africa 171 Andorra 971.54 Europe 172 Papua New Guinea 963.63 Oceania 173 St. Vincent / Grenadines 961.92 North America 174 Grenada 955.44 North America 175 Nepal 939.20 Asia 176 Chad 935.41 Africa 177 Barbados 933.30 North America 178 Belize 925.02 North America 179 Mauritius 923.46 Africa 180 Cambodia 920.00 Asia 181 Montserrat 914.26 North America 182 Bhutan 904.06 Asia 183 Dominica 900.00 North America 184 Brunei 899.87 Asia 185 Bangladesh 898.81 Asia 186 Laos 890.08 Asia 187 American Samoa 883.17 Oceania 188 Mongolia 879.75 Asia 189 São Tomé e Príncipe 878.09 Africa 190 Cook Islands 877.53 Oceania 191 Djibouti 877.37 Africa 192 Samoa 876.41 Oceania 193 Macau 868.79 Asia 194 Cayman Islands 866.71 North America 195 Aruba 858.00 North America 196 Gibraltar 848.58 Europe 197 Timor-Leste 846.80 Asia 198 Pakistan 842.59 Asia 199 Tonga 835.64 Oceania 200 Sri Lanka 832.80 Asia 201 Seychelles 831.19 Africa 202 Somalia 829.81 Africa 203 Guam 824.65 Oceania 204 Liechtenstein 821.71 Europe 205 Bahamas 818.58 North America 206 Turks and Caicos Islands 803.98 North America 207 British Virgin Islands 780.30 North America 208 US Virgin Islands 779.71 North America 209 Anguilla 769.31 North America 210 San Marino 747.42 Europe. Convert this table into latex code Output:

\documentclass{article}
\usepackage{longtable} % For automatically splitting long tables across pages
\begin{document}

\begin{center}
\begin{longtable}{r l r l}
\caption{FIFA-Like Ranking by Continent} \label{tab:ranking} \\
\hline
\textbf{Rank} & \textbf{Country} & \textbf{Points} & \textbf{Continent} \\
hline
\endfirsthead

```
\multicolumn{4}{c}%
{{\bfseries \tablename\ \thetable{} -- continued from previous page}} \\
\hline
\textbf{Rank} & \textbf{Country} & \textbf{Points} & \textbf{Continent} \\
\hline
\endhead
```

\hline \endlastfoot

```
1 & Argentina & 1867.25 & South America \\
2 & France & 1859.78 & Europe \\
3 & Spain & 1853.27 & Europe \\
4 & England & 1813.81 & Europe \\
5 & Brazil & 1775.85 & South America \\
6 & Portugal & 1756.12 & Europe \\
7 & Netherlands & 1747.55 & Europe \\
8 & Belgium & 1740.62 & Europe \\
9 & Italy & 1731.51 & Europe \\
10 & Germany & 1703.79 & Europe \\
11 & Uruguay & 1695.91 & South America \\
12 & Colombia & 1694.44 & South America \\
13 & Croatia & 1691.59 & Europe \\
14 & Morocco & 1688.18 & Africa \\
15 & Japan & 1652.79 & Asia \\
16 & United States & 1645.48 & North America \\
17 & Senegal & 1637.25 & Africa \\
18 & IR Iran & 1635.31 & Asia \\
19 & Mexico & 1627.40 & North America \\
20 & Switzerland & 1625.16 & Europe \\
21 & Denmark & 1611.49 & Europe \\
22 & Austria & 1589.84 & Europe \\
23 & Korea Republic & 1585.45 & Asia \\
24 & Ecuador & 1560.13 & South America \\
25 & Ukraine & 1554.94 & Europe \\
26 & Australia & 1544.15 & Oceania \\
27 & Sweden & 1540.20 & Europe \\
28 & T\"urkiye & 1537.24 & Europe \\
29 & Wales & 1534.39 & Europe \\
30 & Hungary & 1517.57 & Europe \\
31 & Canada & 1515.96 & North America \\
32 & Serbia & 1514.28 & Europe \\
33 & Egypt & 1513.48 & Africa \\
34 & Russia & 1512.32 & Europe \\
35 & Poland & 1510.62 & Europe \\
36 & Panama & 1504.39 & North America \\
37 & Algeria & 1495.85 & Africa \\
38 & Romania & 1494.20 & Europe \\
39 & Greece & 1489.88 & Europe \\
40 & Peru & 1488.42 & South America \\
41 & Slovakia & 1486.19 & Europe \\
42 & Czechia & 1484.88 & Europe \\
43 & Norway & 1484.44 & Europe \\
44 & Nigeria & 1482.23 & Africa \\
45 & Scotland & 1480.24 & Europe \\
46 & C\^ote d'Ivoire & 1478.47 & Africa \\
47 & Venezuela & 1475.71 & South America \\
48 & Qatar & 1474.60 & Asia \\
49 & Cameroon & 1471.49 & Africa \\
```

50 & Chile & 1469.44 & South America \\ 51 & Mali & 1466.10 & Africa \\ 52 & Tunisia & 1464.42 & Africa \\ 53 & Paraguay & 1458.26 & South America \\ 54 & Costa Rica & 1454.99 & North America \\ 55 & Slovenia & 1454.25 & Europe \\ 56 & Iraq & 1442.86 & Asia \\ 57 & South Africa & 1428.87 & Africa \\ 58 & Uzbekistan & 1424.93 & Asia \\ 59 & Saudi Arabia & 1406.56 & Asia \\ 60 & Republic of Ireland & 1400.22 & Europe \\ 61 & Congo DR & 1395.77 & Africa \\ 62 & Jamaica & 1390.85 & North America \\ 63 & UAE & 1385.57 & Asia \\ 64 & Jordan & 1375.91 & Asia \\ 65 & Albania & 1375.57 & Europe \\ 66 & Burkina Faso & 1373.76 & Africa \\ 67 & North Macedonia & 1368.96 & Europe \\ 68 & Georgia & 1362.21 & Europe \\ 69 & Finland & 1361.53 & Europe \\ 70 & Iceland & 1355.10 & Europe \\ 71 & Northern Ireland & 1349.58 & Europe \\ 72 & Cabo Verde & 1328.51 & Africa \\ 73 & Montenegro & 1326.76 & Europe \\ 74 & Bosnia-Herzegovina & 1326.69 & Europe \\ 75 & Honduras & 1325.93 & North America $\backslash\backslash$ 76 & Israel & 1322.25 & Europe (UEFA) \\ 77 & Ghana & 1320.52 & Africa \\ 78 & Guinea & 1311.88 & Africa \\ 79 & Bolivia & 1308.58 & South America \\ 80 & Oman & 1306.67 & Asia \\ 81 & Bahrain & 1305.52 & Asia \\ 82 & Bulgaria & 1301.04 & Europe \\ 83 & El Salvador & 1300.81 & North America \\ 84 & Gabon & 1297.87 & Africa \\ 85 & Angola & 1296.48 & Africa \\ 86 & Haiti & 1287.75 & North America \\ 87 & Zambia & 1279.24 & Africa \\ 88 & Uganda & 1270.42 & Africa \\ 89 & New Zealand & 1269.83 & Oceania \\ 90 & China PR & 1266.38 & Asia \\ 91 & Cura\c{c}ao & 1264.78 & North America \\ 92 & Luxembourg & 1256.42 & Europe \\ 93 & Equatorial Guinea & 1256.20 & Africa \\ 94 & Benin & 1250.99 & Africa \\ 95 & Syria & 1249.47 & Asia \\ 96 & Mozambique & 1232.70 & Africa \\ 97 & Thailand & 1231.17 & Asia \\ 98 & Belarus & 1226.31 & Europe \\

```
99 & Kosovo & 1219.82 & Europe \\
100 & Armenia & 1219.15 & Europe \\
101 & Palestine & 1215.87 & Asia \\
102 & Trinidad and Tobago & 1214.81 & North America \\
103 & Comoros & 1208.99 & Africa \\
104 & Tajikistan & 1203.08 & Asia \\
105 & Guatemala & 1201.35 & North America \\
106 & Tanzania & 1199.33 & Africa \\
107 & Kyrgyz Republic & 1194.19 & Asia \\
108 & Kenya & 1189.65 & Africa \\
109 & Mauritania & 1183.16 & Africa \\
110 & Kazakhstan & 1180.18 & Europe \\
111 & Namibia & 1179.26 & Africa \\
112 & Lebanon & 1171.43 & Asia \\
113 & Sudan & 1165.41 & Africa \\
114 & Vietnam & 1164.79 & Asia \\
115 & Korea DPR & 1164.41 & Asia \\
116 & Madagascar & 1162.02 & Africa \\
117 & Azerbaijan & 1158.70 & Europe \\
118 & Libya & 1158.03 & Africa \\
119 & Togo & 1152.18 & Africa \\
120 & Guinea-Bissau & 1151.72 & Africa \\
121 & Zimbabwe & 1151.06 & Africa \\
122 & Niger & 1145.96 & Africa \\
123 & Estonia & 1142.53 & Europe \\
124 & Rwanda & 1136.06 & Africa \\
125 & Gambia & 1134.69 & Africa \\
126 & India & 1133.62 & Asia \\
127 & Indonesia & 1133.41 & Asia \\
128 & Congo & 1132.65 & Africa \\
129 & Sierra Leone & 1132.06 & Africa \\
130 & Cyprus & 1131.32 & Europe \\
131 & Malawi & 1129.31 & Africa \\
132 & Malaysia & 1115.51 & Asia \\
133 & Nicaragua & 1114.28 & North America \\
134 & Kuwait & 1108.72 & Asia \\
135 & Central African Republic & 1102.23 & Africa \\
136 & Botswana & 1098.85 & Africa \\
137 & Faroe Islands & 1096.88 & Europe \\
138 & Suriname & 1096.60 & South America \\
139 & Burundi & 1087.26 & Africa \\
140 & Latvia & 1084.11 & Europe \\
141 & Dominican Republic & 1083.36 & North America \\
142 & Lithuania & 1069.96 & Europe \\
143 & Turkmenistan & 1065.42 & Asia \\
144 & St. Kitts and Nevis & 1064.99 & North America \\
145 & Liberia & 1063.04 & Africa \\
146 & Ethiopia & 1059.94 & Africa \\
147 & Solomon Islands & 1051.00 & Oceania \backslash\backslash
```

148 & Fiji & 1048.48 & Oceania \\ 149 & Lesotho & 1046.76 & Africa \\ 150 & Philippines & 1046.67 & Asia \\ 151 & Moldova & 1045.59 & Europe \\ 152 & New Caledonia & 1034.43 & Oceania \\ 153 & Tahiti & 1031.47 & Oceania \\ 154 & Guyana & 1030.60 & South America \\ 155 & Hong Kong, China & 1028.76 & Asia \\ 156 & Afghanistan & 1025.00 & Asia \\ 157 & Puerto Rico & 1022.83 & North America \\ 158 & Yemen & 1018.27 & Asia \\ 159 & Eswatini & 1015.25 & Africa \\ 160 & Singapore & 1011.68 & Asia \\ 161 & Antigua and Barbuda & 1010.97 & North America \\ 162 & Maldives & 1001.39 & Asia \\ 163 & Cuba & 994.43 & North America \\ 164 & Vanuatu & 994.22 & Oceania \\ 165 & Chinese Taipei & 990.56 & Asia \\ 166 & Bermuda & 988.19 & North America \\ 167 & St. Lucia & 985.72 & North America \\ 168 & Malta & 983.95 & Europe \\ 169 & Myanmar & 981.35 & Asia \\ 170 & South Sudan & 973.55 & Africa \\ 171 & Andorra & 971.54 & Europe \\ 172 & Papua New Guinea & 963.63 & Oceania \\ 173 & St. Vincent / Grenadines & 961.92 & North America \\ 174 & Grenada & 955.44 & North America \\ 175 & Nepal & 939.20 & Asia \\ 176 & Chad & 935.41 & Africa \\ 177 & Barbados & 933.30 & North America \\ 178 & Belize & 925.02 & North America \\ 179 & Mauritius & 923.46 & Africa \\ 180 & Cambodia & 920.00 & Asia \\ 181 & Montserrat & 914.26 & North America \\ 182 & Bhutan & 904.06 & Asia \\ 183 & Dominica & 900.00 & North America \\ 184 & Brunei & 899.87 & Asia \\ 185 & Bangladesh & 898.81 & Asia \\ 186 & Laos & 890.08 & Asia \\ 187 & American Samoa & 883.17 & Oceania \\ 188 & Mongolia & 879.75 & Asia \\ 189 & S\~ao Tom\'e e Pr\'incipe & 878.09 & Africa \\ 190 & Cook Islands & 877.53 & Oceania \\ 191 & Djibouti & 877.37 & Africa \\ 192 & Samoa & 876.41 & Oceania \\ 193 & Macau & 868.79 & Asia \\ 194 & Cayman Islands & 866.71 & North America \\ 195 & Aruba & 858.00 & North America \\ 196 & Gibraltar & 848.58 & Europe \\

```
197 & Timor-Leste & 846.80 & Asia \\
198 & Pakistan & 842.59 & Asia \\
199 & Tonga & 835.64 & Oceania \\
200 & Sri Lanka & 832.80 & Asia \\
201 & Seychelles & 831.19 & Africa \\
202 & Somalia & 829.81 & Africa \\
203 & Guam & 824.65 & Oceania \\
204 & Liechtenstein & 821.71 & Europe \\
205 & Bahamas & 818.58 & North America \\
206 & Turks and Caicos Islands & 803.98 & North America \\
207 & British Virgin Islands & 780.30 & North America \\
208 & US Virgin Islands & 779.71 & North America \\
209 & Anguilla & 769.31 & North America \\
210 & San Marino & 747.42 & Europe \\
\end{longtable}
\end{center}
```

 $\end{document}$

- OpenAI ChatGPT (Dec 5, 2024 version, ChatGPT-o1) Query: Briefly explain incremental min-max estimation for normalisation Output: Idea: Keep track of running estimates of the minimum and maximum as new data flows in. Method:
 - (a) Initialize running estimates for min_{val} and max_{val} from your first data batch (or some small bootstrap sample).
 - (b) For each new sample x:
 - Update $\min_{val} = \min(\min_{val}, x)$
 - Update $\max_{val} = \max(\max_{val}, x)$
 - (c) Rescale your data as:

$$x_{\text{scaled}} = \frac{x - \min_{\text{val}}}{(\max_{\text{val}} - \min_{\text{val}}) + \epsilon},$$

where ϵ is a small number to avoid division by zero if $\max_{val} \approx \min_{val}$.

Challenge: As new min or max values are discovered, the scale changes, which can shift your previously scaled data. If you can afford some "re-scaling" overhead (e.g., in reinforcement learning or iterative scoring), this is often acceptable. If not, you might have to freeze the parameters after a certain stage or use bounding/clamping strategies.

3. OpenAI *ChatGPT* (Dec 5, 2024 version, ChatGPT-o1)

Query: Explain different methods of local and global sensitivity analysis

Output: There are many methods for sensitivity analysis, each with different strengths and complexity. Broadly, methods can be categorized into local and global approaches: A. Local Sensitivity Analysis How it works: Local methods measure the effect on the model output by making small changes around a specific point in parameter space (often the nominal or baseline point). Typical steps: Partial derivatives / Jacobian analysis: Compute partial derivatives of the model output with respect to each parameter at the baseline point. Larger magnitudes of partial derivatives suggest that the output is more sensitive to changes in those parameters. One-at-a-time (OAT): Vary each parameter slightly from its nominal value while holding other parameters constant, then record the change in the output. Pros: Simple to implement. Provides quick estimates of which parameters are most influential at a specific point. Cons: Only valid around a single point or scenario. May miss interactions between parameters. B. Global Sensitivity Analysis How it works: Global methods assess sensitivity across the entire parameter space (or a significant portion of it), rather than around a single point. They can capture parameter interactions and non-linear effects. Common global methods: Variance-based methods (Sobol' indices): Decompose the variance of the model output among the input parameters. Morris screening (Method of Morris): A more qualitative screening approach that systematically varies parameters and measures the elementary effects on the model output. Regression or correlation-based methods: For example, partial correlation coefficients or partial rank correlation coefficients (PRCC). Pros: Gives insights into overall (global) influence of parameters, including interactions. Cons: Can be computationally expensive if the model is complex or high-dimensional. Requires sampling strategies (like Monte Carlo, Latin Hypercube Sampling, etc.) to explore parameter space thoroughly.

4. OpenAI ChatGPT (May 13, 2024 version, GPT-40)

Query: List statistics for the following countries' men's national football (soccer) teams in the FIFA World Cup or any international tournament/qualification process. For positive stats, I want avg. goals per match, avg. points per match, games played, shots on goal and pass completions. For negative stats, I want avg goals against per match, number of yellow cards, fouls committed per match, offsides, and number of times possession was lost. The list of countries are: Japan, Iran, Korea, Qatar, Iraq, Uzbekistan, Morocco, Senegal, Egypt, France, Spain, England, Portugal, Netherlands, Belgium, Italy, Germany, Croatia, United States, Mexico, Argentina, Brazil, Australia, New Zealand. Compiling comprehensive statistics for men's national football teams across all international tournaments and qualification processes is challenging due to the vast amount of data and varying record-keeping standards across different competitions. However, I can provide an overview of the FIFA World Cup performances for the countries you've listed, focusing on metrics such as average goals per match, games played, and notable achievements.

1. Japan

World Cup Appearances: 7 (1998–2022) Best Performance: Round of 16 (2002, 2010, 2018, 2022) Total Matches Played: 24 Total Goals Scored: 22 Average Goals per Match: 0.92

2. Iran

World Cup Appearances: 6 (1978–2022) Best Performance: Group Stage Total Matches Played: 18 Total Goals Scored: 9 Average Goals per Match: 0.5

3. South Korea

World Cup Appearances: 11 (1954–2022) Best Performance: 4th Place (2002) Total Matches Played: 40 Total Goals Scored: 37 Average Goals per Match: 0.93

4. Qatar

World Cup Appearances: 1 (2022) Best Performance: Group Stage Total Matches Played: 3 Total Goals Scored: 1 Average Goals per Match: 0.33

5. Iraq

World Cup Appearances: 1 (1986) Best Performance: Group Stage Total Matches Played: 3

Total Goals Scored: 1 Average Goals per Match: 0.33

6. Uzbekistan World Cup Appearances: 0

7. Morocco

World Cup Appearances: 6 (1970–2022) Best Performance: 4th Place (2022) Total Matches Played: 23 Total Goals Scored: 20 Average Goals per Match: 0.87

8. Senegal

World Cup Appearances: 3 (2002, 2018, 2022) Best Performance: Quarter-finals (2002) Total Matches Played: 11 Total Goals Scored: 14 Average Goals per Match: 1.27

9. Egypt

World Cup Appearances: 3 (1934, 1990, 2018) Best Performance: Group Stage Total Matches Played: 7 Total Goals Scored: 5 Average Goals per Match: 0.71

10. France

World Cup Appearances: 16 (1930–2022) Best Performance: Champions (1998, 2018) Total Matches Played: 73 Total Goals Scored: 136 Average Goals per Match: 1.86

11. Spain

World Cup Appearances: 16 (1934–2022) Best Performance: Champions (2010) Total Matches Played: 69 Total Goals Scored: 106 Average Goals per Match: 1.54

12. England

World Cup Appearances: 16 (1950–2022) Best Performance: Champions (1966) Total Matches Played: 74 Total Goals Scored: 104 Average Goals per Match: 1.41

13. Portugal

World Cup Appearances: 8 (1966–2022) Best Performance: 3rd Place (1966) Total Matches Played: 34 Total Goals Scored: 66 Average Goals per Match: 1.94

14. Netherlands

World Cup Appearances: 11 (1934–2022) Best Performance: Runners-up (1974, 1978, 2010) Total Matches Played: 55 Total Goals Scored: 100 Average Goals per Match: 1.82

15. Belgium

World Cup Appearances: 14 (1930–2022) Best Performance: Third Place (2018) Total Matches Played: 50 Total Goals Scored: 71 Average Goals per Match: 1.42

16. Italy

World Cup Appearances: 18 (1934–2014) Best Performance: Champions (1934, 1938, 1982, 2006) Total Matches Played: 83 Total Goals Scored: 128 Average Goals per Match: 1.54

17. Germany

World Cup Appearances: 20 (1934–2022) Best Performance: Champions (1954, 1974, 1990, 2014) Total Matches Played: 112 Total Goals Scored: 232 Average Goals per Match: 2.07

18. Croatia

World Cup Appearances: 6 (1998–2022) Best Performance: Runners-up (2018) Total Matches Played: 30 Total Goals Scored: 61 Average Goals per Match: 2.03

19. United States

World Cup Appearances: 11 (1930–2022) Best Performance: Third Place (1930) Total Matches Played: 37 Total Goals Scored: 37 Average Goals per Match: 1.00

20. Mexico

World Cup Appearances: 17 (1930–2022) Best Performance: Quarter-finals (1970, 1986) Total Matches Played: 60 Total Goals Scored: 63 Average Goals per Match: 1.05

21. Argentina

World Cup Appearances: 18 (1930–2022) Best Performance: Champions (1978, 1986, 2022) Total Matches Played: 88 Total Goals Scored: 152 Average Goals per Match: 1.73

22. Brazil

World Cup Appearances: 22 (1930–2022) Best Performance: Champions (1958, 1962, 1970, 1994, 2002) Total Matches Played: 114 Total Goals Scored: 237 Average Goals per Match: 2.08

23. Australia

World Cup Appearances: 6 (1974–2022) Best Performance: Round of 16 (2006, 2022) Total Matches Played: 18 Total Goals Scored: 15 Average Goals per Match: 0.83

24. New Zealand

World Cup Appearances: 2 (1982, 2010) Best Performance: Group Stage Total Matches Played: 6 Total Goals Scored: 4 Average Goals per Match: 0.67

8 Appendix

8.1 Appendix A: Data for Team Selection

Continent	Population (Millions)
Asia	4800
Africa	1400
Europe	750
North America	600
South America	440
Oceania	44

Table 14: Total Populations of Continents (2023)

Continent	Number of Teams
Europe	26
South America	7
Africa	6
Asia	4
North America	4

Table 15: Number of Teams in FIFA Top 48 by Continent (2024)

8.2 Appendix B: Random Index Values

Matrix Size	\mathbf{RI}
1	0.00
2	0.00
3	0.58
4	0.90
5	1.12
6	1.24
7	1.32
8	1.41
9	1.45
10	1.49

Table 16: Random Index Values [2]

8.3 Appendix C: Selective Annealing Program Code

```
1 import random
 2 import numpy as np
 3 import matplotlib.pyplot as plt
 4 import math
 5
6
7 class SportsScheduler:
           def __init__(self, teams, rounds, travel_matrix, x, y, maxtemp, mintemp,
 8
          coe_weights):
                   self.teams = teams
 9
                   self.rounds = rounds
                   self.travel_matrix = travel_matrix
11
                   self.half_schedule = self.initialize_schedule()
                   self.schedule = self.half_schedule + [[(b, a) for a, b in round_matches]
13
         for round_matches in self.half_schedule]
                   self.x = x
14
                   self.y = y
                   self.maxtemp = maxtemp
16
                   self.mintemp = mintemp
                   self.coe_weights = coe_weights
18
19
           def initialize_schedule(self):
20
                   """Generates an initial feasible double round-robin schedule."""
21
                   team_number = self.teams
                   teams = list(range(self.teams))
23
                   opponent_team = 0
24
                   schedule = [[(0, 21), (20, 1), (2, 19), (18, 3), (4, 17), (16, 5), (6, 15),
25
            (14, 7), (8, 13), (12, 9), (10, 11)],
      [(21, 1), (0, 2), (3, 20), (19, 4), (5, 18), (17, 6), (7, 16), (15, 8), (9, 14),
26
          (13, 10), (11, 12)],
      [(2, 21), (1, 3), (4, 0), (20, 5), (6, 19), (18, 7), (8, 17), (16, 9), (10, 15),
27
          (14, 11), (12, 13)],
      [(21, 3), (2, 4), (5, 1), (0, 6), (7, 20), (19, 8), (9, 18), (17, 10), (11, 16),
          (15, 12), (13, 14)],
      [(4, 21), (3, 5), (6, 2), (1, 7), (8, 0), (20, 9), (10, 19), (18, 11), (12, 17),
29
          (16, 13), (14, 15)],
      [(21, 5), (4, 6), (7, 3), (2, 8), (9, 1), (0, 10), (11, 20), (19, 12), (13, 18),
30
          (17, 14), (15, 16)],
      [(6, 21), (5, 7), (8, 4), (3, 9), (10, 2), (1, 11), (12, 0), (20, 13), (14, 19),
31
          (18, 15), (16, 17)],
      [(21, 7), (6, 8), (9, 5), (4, 10), (11, 3), (2, 12), (13, 1), (0, 14), (15, 20),
32
          (19, 16), (17, 18)],
     [(8, 21), (7, 9), (10, 6), (5, 11), (12, 4), (3, 13), (14, 2), (1, 15), (16, 0),
33
          (20, 17), (18, 19)],
     [(21, 9), (8, 10), (11, 7), (6, 12), (13, 5), (4, 14), (15, 3), (2, 16), (17, 1),
34
          (0, 18), (19, 20)],
      [(10, 21), (9, 11), (12, 8), (7, 13), (14, 6), (5, 15), (16, 4), (3, 17), (18, 2),
            (1, 19), (20, 0)],
      [(21, 11), (10, 12), (13, 9), (8, 14), (15, 7), (6, 16), (17, 5), (4, 18), (19, 3)
36
          , (2, 20), (0, 1)],
      [(12, 21), (11, 13), (14, 10), (9, 15), (16, 8), (7, 17), (18, 6), (5, 19), (20, 10)]
37
         4), (3, 0), (1, 2)],
      [(21, 13), (12, 14), (15, 11), (10, 16), (17, 9), (8, 18), (19, 7), (6, 20), (0, 10)]
38
         5), (4, 1), (2, 3)],
      [(14, 21), (13, 15), (16, 12), (11, 17), (18, 10), (9, 19), (20, 8), (7, 0), (1, 10), (10, 10), (10, 10), (10, 10), (10, 10), (11, 10), (10, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10), (11, 10)
39
         6), (5, 2), (3, 4)],
    [(21, 15), (14, 16), (17, 13), (12, 18), (19, 11), (10, 20), (0, 9), (8, 1), (2,
40
```

```
7), (6, 3), (4, 5)],
     [(16, 21), (15, 17), (18, 14), (13, 19), (20, 12), (11, 0), (1, 10), (9, 2), (3, 10)]
41
         8), (7, 4), (5, 6)],
      [(21, 17), (16, 18), (19, 15), (14, 20), (0, 13), (12, 1), (2, 11), (10, 3), (4, 10)]
42
         9), (8, 5), (6, 7)],
      [(18, 21), (17, 19), (20, 16), (15, 0), (1, 14), (13, 2), (3, 12), (11, 4), (5, 12), (11, 4), (5, 12), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (11, 14), (
43
         10), (9, 6), (7, 8)],
      [(21, 19), (18, 20), (0, 17), (16, 1), (2, 15), (14, 3), (4, 13), (12, 5), (6, 11)
44
          , (10, 7), (8, 9)],
      [(20, 21), (19, 0), (1, 18), (17, 2), (3, 16), (15, 4), (5, 14), (13, 6), (7, 12),
45
           (11, 8), (9, 10)]]
46
                  return schedule
47
48
           def compute_travel_distance(self): # maxtemp and mintemp are arrays for max and
49
           min values for all 3, normalised the distance that each team travelled, and
         multiplied final result by the standard dev of normalised travel distance
                   """Computes total travel distance for each team."""
50
                   total_distance = 0
                   tt = 0 # CAN WE REMOVE THIS
52
                  self.maxtemp[0] = 0
53
                  self.mintemp[0] = 10**31
54
                  data = []
                  for team in range(self.teams):
56
                          team_distance = 0
57
                          last_location = team # Assume each team starts at its home location
58
59
                          for r in range(self.rounds):
                                  for home, away in self.schedule[r]:
61
                                         if team == away and team < 20 and home < 20 and away < 20:
                                                 # Away team travels to home team's stadium
63
                                                 team_distance += self.travel_matrix[last_location][home]
                                                 last_location = home # Update last location
65
                                         elif team == home and team < 20 and home < 20 and away < 20:
                                                 # Home team returns to its own stadium if it was away
67
         previously
                                                 team_distance += self.travel_matrix[last_location][home]
                                                 last_location = home # Reset last location to home
69
                          tt += team_distance # CAN WE REMOVE THIS
71
                          self.maxtemp[0] = max(self.maxtemp[0], team_distance)
73
                          self.mintemp[0] = min(self.mintemp[0], team_distance)
74
                          normalised_team_distance = (team_distance - self.mintemp[0])/(self.
75
         maxtemp[0] - self.mintemp[0] + 10**-4) # Min-Max normalisation
                          data.append(normalised_team_distance)
76
                          total_distance += normalised_team_distance # Sum all team distances
77
                   return total_distance * math.sqrt(sum(pow((x - sum(data))/len(data), 2) for
78
           x in data) / len(data)) # D_t * D_sd
79
           def compute_breaks(self): #same as above, but with normalised team breaks
80
81
                   """Computes the number of breaks in the schedule."""
                   total_breaks = 0
82
                   unormalisedbreaks = 0 # Actual number of breaks (just for viewing sake)
83
                   self.maxtemp[1] = 0
84
                   self.mintemp[1] = 10**31
85
                  data = []
86
87
                  for team in range(self.teams):
88
```

```
team_breaks = 0
89
               for r in range(self.rounds - 1):
90
                    home_prev, away_prev = None, None
91
                    home_curr, away_curr = None, None
                    for match in self.schedule[r]:
94
                        if team in match and team < 20:
95
                            home_prev, away_prev = match
96
                            break
97
                    for match in self.schedule[r + 1]:
98
                        if team in match and team < 20:
99
                            home_curr, away_curr = match
100
                            break
                    if home_prev == team and home_curr == team:
                        team_breaks += 1
                    elif away_prev == team and away_curr == team:
                        team_breaks += 1
106
               self.maxtemp[1] = max(self.maxtemp[1], team_breaks)
               self.mintemp[1] = min(self.mintemp[1], team_breaks)
108
               normalised_team_breaks = (team_breaks - (3*self.teams - 6))/(self.
109
      maxtemp[1] - (3*self.teams - 6) + 10**-4)
               data.append(normalised_team_breaks)
110
               total_breaks += normalised_team_breaks
111
               unormalisedbreaks += team_breaks
112
           return total_breaks * math.sqrt(sum(pow((x - sum(data))/len(data), 2) for x
113
       in data) / len(data))
114
       def compute_coe(self): # NEED TO ADD NORMALISATION AND ALSO STANDARD DEVIATION
115
           total_coe = 0
           unormalisedcoe = 0 # Actual number of breaks (just for viewing sake)
117
           self.maxtemp[2] = 0
118
           self.mintemp[2] = 10**31
119
           data= []
120
           coe_matrix = [[0 for i in range(self.teams)] for j in range(self.teams)]
121
           for team in range(self.teams):
               for r in range(self.rounds - 1):
123
                    for i in self.schedule[r]:
                        if i[0] == team and team < 20:</pre>
                            opps1 = i[1]
126
127
                            break
                        elif i[1] == team and team < 20:</pre>
128
                            opps1 = i[0]
                            break
130
                    for i in self.schedule[r+1]:
                        if i[0] == team and team < 20:
                            opps2 = i[1]
133
                            break
134
                        elif i[1] == team and team < 20:</pre>
                            opps2 = i[0]
136
                            break
138
                    if opps1 < 20 and opps2 < 20 and team < 20:
                        coe_matrix[opps2][opps1] += self.coe_weights[opps2][opps1]
139
           for i, j in enumerate(coe_matrix): # the total coe is calculated at the
140
      very end
               team_coe = 0
141
               for k in j:
142
143
                    team_coe += k**2
                    self.maxtemp[2] = max(self.maxtemp[2], k**2)
144
```

```
self.mintemp[2] = min(self.mintemp[2], k**2)
145
                   normalised_coe = (team_coe - self.mintemp[2])/(self.maxtemp[2] -
146
      self.mintemp[2] + 10**-4)
                   data.append(normalised_coe)
147
                   total_coe += normalised_coe
148
                   unormalisedcoe += k**2
149
           return (total_coe) * math.sqrt(sum(pow((x - sum(data))/len(data), 2) for x
      in data) / len(data))
      def perturb_schedule(self):
           r1 = random.randint(0, self.rounds // 2 - 1) # Select a random round
           r2 = random.randint(0, self.rounds // 2 - 1)
           while r1 == r2:
               r1 = random.randint(0, self.rounds //2 - 1) # Select a random round
               r2 = random.randint(0, self.rounds // 2 - 1)
158
           """match_idx1, match_idx2 = random.sample(range(len(self.half_schedule[r]))
159
      , 2)
           # Extract matches
160
           (teamA1, teamB1) = self.half_schedule[r][match_idx1]
161
           (teamA2, teamB2) = self.half_schedule[r][match_idx2]"""
162
           rando = random.random()
163
           # Randomly choose whether to swap home teams or away teams
           if rando <= 0.5:
               temp = self.half_schedule[r1]
               self.half_schedule[r1] = self.half_schedule[r2]
167
               self.half_schedule[r2] = temp
168
           # **Randomly flip home/away for a match**
170
           elif rando > 0.5: # 50% chance of flipping home/away
171
               flip_idx = random.randint(0, len(self.half_schedule[r1]) - 1) # Select
       a random match
               home, away = self.half_schedule[r1][flip_idx]
173
               self.half_schedule[r1][flip_idx] = (away, home) # Swap home and away
174
      teams
      def optimize_schedule(self, iterations=50000, temp=5000, cooling_rate=0.95): #
      ADD THE SCALING FACTORS, MAYBE FINE TUNE THE COOLING RATE?
           current_score = 0.122 * self.compute_travel_distance() + 0.32 * self.
      compute_breaks() + 0.558 * self.compute_coe()
           best_schedule = self.half_schedule
178
           best_score = current_score
179
180
           for i in range(iterations):
181
               prev_half_schedule = self.half_schedule.copy()
182
               prev_schedule = self.schedule.copy() # Save current schedule before
183
      perturbation
               self.perturb_schedule()
184
               mirror_schedule = [[(b, a) for a, b in round_matches] for round_matches
185
       in self.half_schedule] # For mirrored 2RR
               self.schedule = self.half_schedule.copy() + mirror_schedule
186
               new_score = 0.122 * self.compute_travel_distance() + 0.32 * self.
187
      compute_breaks() + 0.558 * self.compute_coe()
               if new_score < current_score or random.uniform(0, 1) < np.exp((
188
      current_score - new_score) / temp):
                   current_score = new_score
189
                   if new_score < best_score:</pre>
190
                       best_score = new_score
191
                       best_schedule = self.half_schedule.copy()
192
```

```
else:
194
                   self.schedule = prev_schedule
195
                   self.half_schedule = prev_half_schedule
196
198
               temp *= cooling_rate
199
               #print(f"Index: {i}, BEST SCORE: {best_score}, NEW SCORE: {new_score},
200
      TRAVEL DIST: {self.compute_travel_distance()}, BREAKS: {self.compute_breaks()},
      COE: {self.compute_coe()}")
               self.x.append(i)
201
               self.y.append(new_score)
202
           self.schedule = best_schedule + [[(b, a) for a, b in round_matches] for
203
      round_matches in best_schedule]
204
      def print_schedule(self):
205
           for r, matches in enumerate(self.schedule):
206
               print(f"Round {r + 1}: {matches}")
207
208
209
_{210} maxtemp = [0, 0, 0]
_{211} mintemp = [2^31, 2^31, 2^31]
212 teams = 22
213 rounds = 42
214 x = []
215 y = []
216 travel_matrix = [[0.0, 1467.0821660993345, 516.3264566563297, 1590.2086898490559,
      1868.5088968693283, 1724.6141703183075, 1691.2985806437446, 11174.566504002463,
      5284.048983656849, 10444.31853182325, 5820.528136037073, 5736.635115217819,
      8685.284193743464, 9564.425398590814, 7686.385025251921, 18179.647894866892,
      19567.557825130145, 586.8297367690218, 2774.308407758066, 3819.079406277047],
      [1467.0821660993345, 0.0, 1055.6346248108925, 348.00627885993003,
      418.98047848391167, 262.15818957348614, 639.4710953881632, 9731.120798166421,
      4206.554167181715, 8978.341702078564, 4975.36962029242, 6175.3455310955715,
      9220.236103435493, 11029.14194668471, 9152.583845717401, 16942.6234396044,
      18971.91090239031, 1895.0867137852083, 4200.457932581317, 3218.5799310676944],
      [516.3264566563297, 1055.6346248108925, 0.0, 1261.6691305711001,
      1473.763938819607, 1317.5396743687882, 1183.0775020283336, 10783.883961531097,
      4772.534349604966, 10010.400869265426, 5336.207374769126, 6098.57027369128,
      9087.584764640113, 10023.024259511596, 8122.977525507234, 17672.656947471856,
      19858.046547601258, 839.5427413964993, 3146.9811973774003, 3360.68282029247],
      [1590.2086898490559, 348.00627885993003, 1261.6691305711001, 0.0,
      368.8080042370103, 330.1534027921841, 967.7762937956412, 9585.237435818703,
      4406.343187031414, 8879.000567350036, 5227.227687931629, 5898.212304223635,
      8942.54989372708, 11095.870043503457, 9252.760946815617, 16985.14712456977,
      18801.786413356574, 2083.44632145609, 4361.801131576963, 3528.2842454123866]
      [1868.5088968693283, 418.98047848391167, 1473.763938819607, 368.8080042370103,
      0.0, 162.12755236185538, 819.4225557354451, 9316.79982604, 4057.16414456093,
      8576.93573922619, 4906.807745283281, 6207.77821311941, 9247.803489753262,
      11417.017580976173, 9553.685709925274, 16626.906108644744, 18559.504178626492,
      2312.8490570759836, 4616.094321056878, 3279.5168741971474], [1724.6141703183075,
       262.15818957348614, 1317.5396743687882, 330.1534027921841, 162.12755236185538,
      0.0, 700.5630275878717, 9469.042667782673, 4079.601694296566, 8719.754759234804,
       4897.4891297109725, 6225.647945918358, 9269.36002518571, 11282.343048503737,
      9410.981180584338, 16727.908007081427, 18710.5107259828, 2157.057584753352,
      4462.609428031834, 3218.6829821946085], [1691.2985806437446, 639.4710953881632,
      1183.0775020283336, 967.7762937956412, 819.4225557354451, 700.5630275878717,
      0.0, 9742.34661383645, 3681.561823242904, 8904.157759415211, 4381.854299790954,
      6800.457057270919, 9843.613094981349, 11164.300991546443, 9239.653776709147,
```

193

16547.49927078005, 18768.864831463434, 1947.8793109632886, 4221.021464857709, 2581.4149726866476], [11174.566504002463, 9731.120798166421, 10783.883961531097, 9585.237435818703, 9316.79982604, 9469.042667782673, 9742.34661383645, 0.0, 7688.676891035909, 1164.5531233189074, 8273.570205947644, 10931.60379022631, 11331.687738535886, 18367.819473730848, 18561.496399315656, 7786.634203464949, 9242.704812623431, 11622.151273182062, 13930.862127338047, 9578.604472406825], [5284.048983656849, 4206.554167181715, 4772.534349604966, 4406.343187031414, 4057.16414456093, 4079.601694296566, 3681.561823242904, 7688.676891035909, 0.0, 6569.965665095427, 1159.8419606504087, 10186.905623329661, 13167.644632968615, 13764.033905353368, 11850.854726989945, 12900.732867211686, 15142.763365614594, 5323.577084137162, 7159.465522647071, 1965.9942231526625], [10444.31853182325, 8978.341702078564, 10010.400869265426, 8879.000567350036, 8576.93573922619, 8719.754759234804, 8904.157759415211, 1164.5531233189074, 6569.965665095427, 0.0, 7115.797755208296, 11194.084959520504, 12085.618895920601, 19431.506960194038, 18130.620279822055, 8293.138810674252, 9996.855956113674, 10834.575918333576, 13124.919685045326, 8489.331642285566], [5820.528136037073, 4975.36962029242, 5336.207374769126, 5227.227687931629, 4906.807745283281, 4897.4891297109725, 4381.854299790954, 8273.570205947644, 1159.8419606504087, 7115.797755208296, 0.0, 11112.77164633971, 14144.283509221788, 13309.07110850156, 11512.26738502865, 12371.475487698728, 14557.158521061714, 5732.637515113728, 7231.427091490372, 2053.5981035996974], [5736.635115217819, 6175.3455310955715, 6098.57027369128, 5898.212304223635, 6207.77821311941, 6225.647945918358, 6800.457057270919, 10931.60379022631, 10186.905623329661, 11194.084959520504, 11112.77164633971, 0.0, 3045.1480424891133, 8351.693151923058, 7692.302998322544, 15726.63447498675, 14075.774744144914, 6108.6719307419, 6366.570942978704, 9368.074900981315], [8685.284193743464, 9220.236103435493, 9087.584764640113, 8942.54989372708, 9247.803489753262, 9269.36002518571, 9843.613094981349, 11331.687738535886, 13167.644632968615, 12085.618895920601, 14144.283509221788, 3045.1480424891133, 0.0, 7353.115216969745, 7663.431357540835, 12978.602606012932, 11084.028159840185, 8983.59798951676, 8632.86620200662, 12402.288551959075], [9564.425398590814, 11029.14194668471, 10023.024259511596, 11095.870043503457, 11417.017580976173 11282.343048503737, 11164.300991546443, 18367.819473730848, 13764.033905353368, 19431.506960194038, 13309.07110850156, 8351.693151923058, 7353.115216969745, 0.0, 1963.5712194407718, 11834.980211873899, 10007.453739817467, 9217.85944045305, 6968.338335585247, 11814.10378533585], [7686.385025251921, 9152.583845717401, 8122.977525507234, 9252.760946815617, 9553.685709925274, 9410.981180584338, 9239.653776709147, 18561.496399315656, 11850.854726989945, 18130.620279822055, 11512.26738502865, 7692.302998322544, 7663.431357540835, 1963.5712194407718, 0.0, 13543.983213972235, 11892.875900556339, 7302.282962863877, 5024.460855155926, 9888.342537814473], [18179.647894866892, 16942.6234396044, 17672.656947471856, 16985.14712456977, 16626.906108644744, $16727.908007081427\,,\ 16547.49927078005\,,\ 7786.634203464949\,,\ 12900.732867211686\,,$ 8293.138810674252, 12371.475487698728, 15726.63447498675, 12978.602606012932, 11834.980211873899, 13543.983213972235, 0.0, 2243.1716864983932, 18056.985430785593, 17598.80887633452, 14396.395286223635], [19567.557825130145, 18971.91090239031, 19858.046547601258, 18801.786413356574, 18559.504178626492, 18710.5107259828, 18768.864831463434, 9242.704812623431, 15142.763365614594, 9996.855956113674, 14557.158521061714, 14075.774744144914, 11084.028159840185, 10007.453739817467, 11892.875900556339, 2243.1716864983932, 0.0, 19125.131777588675, 16838.449211947947, 16520.539650401995], [586.8297367690218, 1895.0867137852083, 839.5427413964993, 2083.44632145609, 2312.8490570759836, 2157.057584753352, 1947.8793109632886, 11622.151273182062, 5323.577084137162, 10834.575918333576, 5732.637515113728, 6108.6719307419, 8983.59798951676, 9217.85944045305, 7302.282962863877, 18056.985430785593, 19125.131777588675, 0.0, 2309.025780825603, 3687.376498134921], [2774.308407758066, 4200.457932581317, 3146.9811973774003, 4361.801131576963, 4616.094321056878, 4462.609428031834, 4221.021464857709, 13930.862127338047, 7159.465522647071, 13124.919685045326, 7231.427091490372, 6366.570942978704, 8632.86620200662,

6968.338335585247, 5024.460855155926, 17598.80887633452, 16838.449211947947, 2309.025780825603, 0.0, 5255.83602592503], [3819.079406277047, 3218.5799310676944, 3360.68282029247, 3528.2842454123866, 3279.5168741971474, 3218.6829821946085, 2581.4149726866476, 9578.604472406825, 1965.9942231526625, 8489.331642285566, 2053.5981035996974, 9368.074900981315, 12402.288551959075, 11814.10378533585, 9888.342537814473, 14396.395286223635, 16520.539650401995, 3687.376498134921, 5255.83602592503, 0.0]] 217 coe_weights = [[0.0, -0.06564060360000001, -0.020365161000000076, -0.5614158869000001, -0.34738277340000007, -0.302861254, -0.44720518690000005, -0.7760801617, -0.29283516910000007, -0.33585504620000006, 0.19230576189999993, 0.1129723654, -0.16430702530000008, -0.43804666690000005, -0.14560999410000008, -0.32277216710000006, 0.20169183179999994, -0.6310633372000001, -0.68835474887, 0], [0.06564060360000001, 0.0, 0.04527544259999994, -0.49577528330000004, -0.28174216980000005, -0.2372206504, -0.38156458330000004, -0.7104395581, -0.22719456550000006, -0.27021444260000005, 0.257946365499999994, 0.178612969, -0.09866642170000006, -0.37240606330000003, -0.07996939050000007, $-0.25713156350000005\,,\ 0.26733243539999996\,,\ -0.5654227336000001\,,\ -0.62271414527\,,$ 0], [0.020365161000000076, -0.04527544259999994, 0.0, -0.5410507258999999, -0.3270176124, -0.2824960929999995, -0.4268400259, -0.7557150007, -0.2724700081, -0.3154898852, 0.2126709229, 0.13333752640000007, -0.1439418643, -0.41768150589999997, -0.1252448331, -0.3024070061, 0.22205699280000002, -0.6106981762, -0.66798958787, 0], [0.5614158869000001, 0.49577528330000004, 0.5410507258999999, 0.0, 0.21403311349999998, 0.2585546329, 0.1142107, -0.2146642748, 0.2685807178, 0.2255608407, 0.7537216488, 0.6743882523, 0.3971088616, 0.12336922, 0.41580589279999997, 0.2386437198, 0.7631077186999999, -0.0696474503, -0.12693886196999998, 0], [0.34738277340000007, 0.28174216980000005, 0.3270176124, -0.21403311349999998, 0.0, $0.04452151940000004\,, \ -0.09982241349999998\,, \ -0.4286973883\,, \ 0.0545476043\,,$ 0.011527727200000004, 0.5396885353, 0.46035513880000006, 0.1830757481, -0.09066389349999998, 0.20177277929999998, 0.024610606300000004, 0.5490746052, -0.28368056379999995, -0.34097197546999997, 0], [0.302861254, 0.2372206504, $0.2824960929999995\,,\ -0.2585546329\,,\ -0.04452151940000004\,,\ 0.0\,,$ -0.14434393290000003, -0.4732189077, 0.010026084899999954, -0.03299379220000004, $0.49516701589999995\,,\ 0.4158336194\,,\ 0.13855422869999995\,,\ -0.13518541290000002\,,$ 0.15725125989999994, -0.01991091310000004, 0.5045530858, -0.3282020832, -0.38549349487, 0], [0.44720518690000005, 0.38156458330000004, 0.4268400259, -0.1142107, 0.09982241349999998, 0.14434393290000003, 0.0, -0.3288749748, 0.15437001779999998, 0.11135014069999999, 0.6395109487999999, 0.5601775523000001, 0.2828981616, 0.00915852000000003, 0.30159519279999997, 0.12443301979999999, 0.6488970187, -0.1838581503, -0.24114956196999998, 0], [0.7760801617, 0.7104395581, 0.7557150007, 0.2146642748, 0.4286973883, 0.4732189077, 0.3288749748, 0, 0.4832449926, 0.4402251155, 0.9683859236, 0.8890525271, 0.6117731364, 0.3380334948, 0.6304701676, 0.4533079946, 0.9777719935, 0.1450168245, 0.08772541283, 0], [0.29283516910000007, 0.22719456550000006, 0.2724700081, -0.2685807178, -0.0545476043, -0.010026084899999954, -0.15437001779999998, -0.4832449926, 0.0, -0.04301987709999999, 0.485140931, 0.40580753450000007, 0.1285281438, -0.14521149779999998, 0.14722517499999999, -0.029936997999999992, 0.4945270009, $-0.3382281681\,,\ -0.39551957976999996\,,\ 0]\,,\ [0.33585504620000006\,,$ 0.27021444260000005, 0.3154898852, -0.2255608407, -0.011527727200000004, 0.03299379220000004, -0.11135014069999999, -0.4402251155, 0.04301987709999999, 0.0, 0.5281608081, 0.44882741160000006, 0.1715480209, -0.10219162069999999, 0.19024505209999998, 0.0130828791, 0.537546878, -0.295208291, -0.35249970266999997, 0], [-0.19230576189999993, -0.25794636549999994, -0.2126709229, -0.7537216488, -0.5396885353, -0.49516701589999995, -0.6395109487999999, -0.9683859236, -0.485140931, -0.5281608081, 0.0, -0.07933339649999993, -0.3566127872, -0.6303524288, -0.337915756, -0.515077929, 0.009386069900000016, -0.8233690991, -0.88066051077, 0], [-0.1129723654, -0.178612969, -0.13333752640000007, -0.6743882523, -0.46035513880000006,

-0.4158336194, -0.5601775523000001, -0.8890525271, -0.40580753450000007, -0.44882741160000006, 0.07933339649999993, 0.0, -0.2772793907000001, -0.5510190323, -0.2585823595000001, -0.43574453250000006, 0.08871946639999995, -0.7440357026000001, -0.80132711427, 0], [0.16430702530000008, 0.09866642170000006, 0.1439418643, -0.3971088616, -0.1830757481, -0.13855422869999995, -0.2828981616, -0.6117731364, -0.1285281438, -0.1715480209, 0.3566127872, 0.2772793907000001, 0.0, -0.27373964159999997, 0.01869703119999999, -0.1584651418, 0.3659988571, -0.4667563119, -0.52404772357, 0], [0.43804666690000005, 0.37240606330000003, 0.41768150589999997, -0.12336922, 0.09066389349999998, 0.13518541290000002, -0.00915852000000003, -0.3380334948, 0.14521149779999998, 0.10219162069999999, 0.6303524288, 0.5510190323, 0.27373964159999997, 0.0, 0.29243667279999996, 0.11527449979999999, 0.6397384987, -0.1930166703, -0.25030808197, 0], [0.14560999410000008, 0.07996939050000007, 0.1252448331, -0.41580589279999997, -0.20177277929999998, -0.15725125989999994, -0.30159519279999997, -0.6304701676, -0.14722517499999999, -0.19024505209999998, 0.337915756, 0.2585823595000001, -0.01869703119999999, -0.29243667279999996, 0.0, -0.17716217299999998, 0.3473018259, -0.4854533431, -0.54274475477, 0], [0.32277216710000006, 0.25713156350000005, 0.3024070061, -0.2386437198, -0.024610606300000004, 0.01991091310000004, -0.12443301979999999, -0.4533079946, 0.029936997999999992, -0.0130828791, 0.515077929, 0.43574453250000006, 0.1584651418, -0.11527449979999999, 0.17716217299999998, 0.0, 0.5244639989, $-0.30829117009999996\,,\ -0.36558258176999997\,,\ 0]\,,\ [-0.20169183179999994\,,$ -0.26733243539999996, -0.22205699280000002, -0.76310771869999999, -0.5490746052, $-0.5045530858\,,\ -0.6488970187\,,\ -0.9777719935\,,\ -0.4945270009\,,\ -0.537546878\,,$ -0.009386069900000016, -0.08871946639999995, -0.3659988571, -0.6397384987, -0.3473018259, -0.5244639989, 0.0, -0.832755169, -0.89004658067, 0], [0.6310633372000001, 0.5654227336000001, 0.6106981762, 0.0696474503, 0.28368056379999995, 0.3282020832, 0.1838581503, -0.1450168245, 0.3382281681, 0.295208291, 0.8233690991, 0.7440357026000001, 0.4667563119, 0.1930166703, 0.4854533431, 0.30829117009999996, 0.832755169, 0.0, -0.05729141167, 0], [0.68835474887, 0.62271414527, 0.66798958787, 0.12693886196999998, 0.34097197546999997, 0.38549349487, 0.24114956196999998, -0.08772541283, 0.39551957976999996, 0.35249970266999997, 0.88066051077, 0.80132711427, 0.52404772357, 0.25030808197, 0.54274475477, 0.36558258176999997, 0.89004658067, 0.05729141167, 0.0, 0], [0.5954082688000001, 0.5297676652000001, 0.5750431078, 0.03399238190000001, 0.2480254954, 0.2925470148, 0.1482030819, -0.1806718929, 0.3025730997, 0.2595532226, 0.7877140307, 0.7083806342000001, 0.4311012435, 0.1573616019, 0.4497982747, 0.27263610169999997, 0.7971001006, -0.03565506839999999, -0.09294648006999999, 0]] 218 scheduler = SportsScheduler(teams, rounds, travel_matrix, x, y, maxtemp, mintemp, coe_weights) 219 scheduler.optimize_schedule() 220 scheduler.print_schedule() 221 plt.figure(dpi=100) 222 plt.rcParams['axes.formatter.useoffset'] = False 223 plt.plot(x,y) 224 plt.xlabel("Iterations") 225 plt.ylabel("Score") 226 plt.show()

8.4 Appendix D: Carry Over Effects Matrix Generator

```
1 coe_weights = [[0 for i in range(20)] for j in range(20)]
_{2} team_strengths = [0.7760801617,
3 0.7104395581,
4 0.7557150007,
<sup>5</sup> 0.2146642748,
6 0.4286973883,
7 0.4732189077,
8 0.3288749748,
90,
10 0.4832449926,
11 0.4402251155,
12 0.9683859236,
13 0.8890525271,
14 0.6117731364,
15 0.3380334948,
16 0.6304701676,
17 0.4533079946,
18 0.9777719935,
19 0.1450168245,
0.08772541283,
0.1806718929,
22 1,
<sup>23</sup> 0.9332477909,
<sup>24</sup> 0.4939079582,
25 0.5536905825]
26 for i, j in enumerate(coe_weights):
      for k in range(len(j)-1):
27
           j[k] = team_strengths[k] - team_strengths[i]
28
29 print(coe_weights)
```