21-393 Operations Research II

17 December 2013

Jae Cho
Kenneth Poon
Vaughn Ridings

Optimizing the Premier League Schedule

Background

The Premier League is the highest professional football league in England that consists of 20 football clubs. It is the primary football competition in England and is also the most watched football league in the world. The competition runs every year from August to May. Each team plays two matches against every team in the league, once at home and once away. Each team therefore plays a total of 38 matches which sums to 380 matches in a league every season.

The competition ranks all 20 teams by the number of points they acquire through the 38 games they play in a season. Each team receives three points for a win, one point for a draw, and zero points for a loss. At the end of the season, the team with the most points is declared the winner. If two or more teams have same number of points, then the league ranks them by goal differential (goals scored minus goals conceded).

Objective

In this project, our goal is to find the optimal schedule for the English Premier League. More precisely, we want to find the most "fair" schedule for all teams in the league. Frequently, there are complaints and concerns with the fairness of the schedule in the Premier League. Some teams argue that they have to travel further in consecutive games and some teams complain about how many days of rest they receive compared to their opponents. Of course, our procedure to make a"fair" schedule will be different from how the league actually schedules the matches. For example, the league has to take into account constraints for other competitive matches, municipal requests to avoid certain days due to other events, commercial concerns, television preferences, and many more.

Data Collected

We collected several data for our optimization model. We obtained the records for all the wins and losses for every match from the previous season, including which team was the home team, which one was the away team, the goals scored by the home team, and the goals conceded by the home team. We also collected data for all the distances between every pair of stadia. The data was derived from the website Football-Data.co.uk which keeps records of every Premier League match back to 1992.

Creating a Model

There are several methods that could be used to help create a "fair" schedule for a league. In our optimization problem, we decided to maximize the schedule advantage for the weakest team first, then the next weakest team, then the next, until we completed the scheduling with the strongest team. To do this, we obviously begin by creating the schedule for the weakest team by maximizing their advantage. Next we construct the schedule for the next weakest team. However, we must keep in mind that this team is constrained by the prior team's schedule. To illustrate this, consider the weakest team to be team A and the next weakest team to be team B. After the first step, we have already set team A's schedule so team B must play team A according to team A's schedule. Furthermore, when scheduling team B's opponents, team A and team B cannot play the same opponent at the same time. This of course places constraints on which opponents team B can play. As mentioned previously, we continue this algorithm until the final (strongest) team. By this process we obtain a full league schedule.

However, we have a problem when it comes to measuring the advantage a team has. First off, what is an "advantage"? What is considered a "good" schedule to have? Does the order of opponents matter? Is it good to pattern your games home and away or is it better to have a run of home games and then a run of away games? Are there other variables that influence advantage when we create a league schedule?

To begin to answer these question, we first define exactly what advantage is. Although there were several valid ways to define it, we label advantage as the expected goal differential a team will have in a game. A goal differential is the number of goals a given team scores in a game minus the number of goals that their opponent scores in a game. Therefore this can be a positive or negative number. To illustrate, if your team wins 3-1, then you win by a +2 goal differential, or by our definition, a +2 goal advantage. The same is if you win 2-0, or 4-2. However, if you lose 5-3, then you will have a -2 goal advantage, as your opponent scored two more goals than you did. Ties, of course, would be a 0 goal advantage. Defining advantage in this way is very useful. It gives us an indication of not only which team is likely to win, but also by how much they will win. In sum, we want to maximize a teams advantage over an entire season. In other words, we want to give them the best strength of schedule.

Now that we have defined advantage, we need to deduce which variables help influence advantage. To help determine this, we used the statistical program R to help us determine exactly what variables change a team's advantage. For simplicity, we only used the data from the 2012/2013 season to help model expected advantage. There were plenty of variables to choose from including the opponents for each game, whether a game was played at home or away, the number of home games in a row or the number of away games in a row, the ranking of the opponent based off their finish from last season, the distance traveled for each team, and the scoreline for every game. From this data we linearly regressed several combinations for variables against expected goal advantage to determine the impact of each variable. In the end, we decided to take the model with the following variables: whether the match was home or away (which we named HomeAwayBin), how many home games a team in question had played up until the current matchday (which we called HomeString), how many away games a team in question had played up until the current match day (called AwayString), and the logarithm of the sum of the ranks of the current opponent and previous

two opponents (called LogRank). The R output is shown below:

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -3.2174     0.6690  -4.809 1.83e-06 ***
HomeAwayBin   0.7456     0.1558   4.787 2.04e-06 ***
HomeString    0.2652     0.1470   1.804   0.0716 .
AwayString    0.1204     0.1501   0.802   0.4226
LogRank       0.7692     0.1897   4.056 5.52e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This regression gives us the following formula we desire:

$$Advantage = 0.7456HomeAwayBin + 0.2652HomeString + 0.1204AwayString + 0.7692LogRank$$

These variables were determined to be the most significant influences on advantage and they give interesting insights. First note that the regression predicts the advantage each team has per game. Since each team plays 38 games per season, each team will have 38 predicted advantage values over a season. The sum of these advantages gives a team its entire season advantage or strength of schedule as defined beforehand. As seen from the R output, the more home or away games a team has in a row, the more significant their game advantage is. Therefore, it is not ideal for a team to play home / away / home / away but rather have a couple home games in a row then a couple away games in a row (i.e. home / home / away / away). This will be explained in more detail later. Furthermore, since we created a linear model, it does not matter when a team plays an opponent at home or when they play them away (in fact, due to the structure of the league, you must play every team home and away by necessity). This renders the HomeAwayBin variable useless for our optimization objectives and thus we do not need to optimize on it. However, how many home and away games you have in a row does matter, so we will still optimize on the HomeString and AwayString variables. Since the model is additive, this means we can schedule the opponents and then schedule when the teams play home or away.

The model does indicate that the order of the teams you play does matter. The variable responsible for this, LogRank, merits closer inspection. Often in sports, it is seen as disadvantageous to play several high-ranked teams in a row. The rankings of teams is determined by the previous season's results. The winner from the previous season would have a ranking of 1, second place would have a ranking of 2, all the way down to the the weakest team have a ranking of 20. The last variable listed above, LogRank, takes the rank of the opponent currently being played plus the ranks of the previous two teams played and then takes the natural logarithm of that sum to obtain LogRank. The R output indicates that having a higher LogRank increases a teams advantage. From here it might seem intuitive that it would be better to play all the weaker teams at once since that would increase the LogRank value and thus increase a teams advantage. While this will indeed increase the team's advantage for the individual game, it will not help the team's strength of schedule for an entire season.

Take the following example as an illustration. Suppose you play teams 17, 18, and 19 (in terms of ranking) all together at one part of the season and teams 1, 2, and 3 in another part

of the season. This means the log rank of the first set will be log(54) and the log rank of the second set will be log(6). Since we sum up each game advantage to obtain the strength of schedule over an entire season, the sum of these will be log(54) + log (6) = 5.7807. However, if we flip team 19 with team 1 such that we play 17, 18, and 1 at one part of a season and then teams 19, 2, and 3 in another part, we obtain the sum log(36) + log(24) = 6.7616 > 5.7807. From this example, we see that optimizing on the log function helps encourage a more balanced schedule in terms of the preseason ranking of teams.

As mentioned before, we can schedule the order of opponents and then the home and away matches separately. Therefore, we can begin constructing a season schedule by denoting the order of opponents for each team. Recall that each team plays each other twice throughout the entire season. To help simplify the problem, we will optimize on the first half of the season where each team has played each other once, then repeat this order for the second half of the season. This technique is frequently used in sports scheduling and is often called "mirroring". Later on when we schedule when teams play home and away, the mirroring technique will again come in handy.

Optimizing the League Schedule

Although in the Premier League there are 19 opponents for each team, we will reduce our scenario down to 10 teams where there are only 9 opponents for each team. This is because our optimization software, standard Microsoft Excel ®, cannot adequately compile a solution for 20 teams. (In fact, Excel approaches the problem quite crudely by trying all n! permutations, seeing which ones violate the constraints, and then taking the maximum value of the ones that do not). Do note however, that the following process is exactly the same for a larger scale problem.

As outline above, we begin with the worst team (the 10^{th} best team according to the end-of-season rankings from last season) and optimize their schedule first. Since we are using mirroring, we only need to schedule the first 9 games. The last 9 games will follow the exact same pattern we established for the first 9 games. Denote each game as G_1, G_2, , G_9. Each $G_i \in \{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8,\ 9\ \}$ and $G_i \neq G_j$ for all $i \neq j$. As detailed above we want to maximize the log sum of the last three opponents for each game. In other words: we want to maximize the value:

$$max\ \ ln(G_1) + ln(G_1 + G_2) + ln(G_1 + G_2 + G_3) + ln(G_2 + G_3 + G_4) + ln(G_3 + G_4 + G_5) + ln(G_4 + G_5 + G_6) + ln(G_5 + G_6 + G_7) + ln(G_6 + G_7 + G_8) + ln(G_7 + G_8 + G_9)$$

$$\text{s.t. } G_i \in \{1, 2, ..., 9\} \text{ for all } i = 1, 2, ..., 9$$
$$G_i \neq G_j \text{ for all } i \neq j$$

Plugging the appropriate constraints into Excel, we obtain this schedule for team 10:

	Matchday								
	1	2	3	4	5	6	7	8	9
Team 10	9	6	3	7	5	4	8	2	1
Team 9									
Team 8									
Team 7									
Team 6									
Team 5									
Team 4									
Team 3									
Team 2									
Team 1									

So team 10 will play teams 9, 6, 3, 7, 5, 4, 8, 2, 1 in that order. Note that team 10 does not ever play 3 difficult opponents in a row, nor do they play three easy opponents in a row, as intended. Moving on, we can now optimize team 9's schedule. However, there are a few more additional constraints for team 9's schedule optimization. First, they must play team 10 the first week. Second, they can not play team 6 the second week, nor can they play team 3 the third week, etc. because team 10 is already playing those opponents. Adding in these constraints, we now optimize the schedule for team 9 to obtain the following:

	Matchday								
	1	2	3	4	5	6	7	8	9
Team 10	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2
Team 8									
Team 7									
Team 6									
Team 5									
Team 4									
Team 3									
Team 2									
Team 1									

So team 9 will play teams 10, 5, 4, 8, 3, 6, 7, 1, 2 in that order. By design, team 9 does not play any of team 10's opponents on the same day and team 9 plays team 10 on the day determined by team 10's schedule optimization. We can now continue with team 8. Again, the constraints increase because team 8 must now play team 10 on matchday 7 and must play team 9 on matchday 4. Of course, team 8 cannot play any of team 9 or 10's opponents on the same matchday. This type of greedy algorithm continues in this vein until we get the following scheduling table:

	Matchday								
	1	2	3	4	5	6	7	8	9
Team 10	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2
Team 8	7	4	2	9	6	1	10	5	3
Team 7	8	2	6	10	1	5	9	3	4
Team 6	3	10	7	1	8	9	2	4	5
Team 5	4	9	1	2	10	7	3	8	6
Team 4	5	8	9	3	2	10	1	6	7
Team 3	6	1	10	4	9	2	5	7	8
Team 2	1	7	8	5	4	3	6	10	9
Team 1	2	3	5	6	7	8	4	9	10

Keep in mind that we use the mirroring technique to schedule the second half of the season as well, so the order of the opponents in the second half of the season will be the same as the first half. Now all we have remaining for the scheduling process is to determine when teams will be at home and when they will be away. Again the mirroring technique will be used, except in this instance, if a team plays at home against an opponent in the first half of the season, that team will play the same opponent away in the second half of the season and vice versa. Here, we will put an additional restriction on scheduling home/away matches that are often used when formatting Premier League seasons. No team may have more than two home or away games in a row throughout the season.

Unlike the scheduling of opponents, scheduling when a team plays home or away can be done manually due to its relative simplicity. Recall that the linear model created earlier has these two terms: $0.2652HomeString + 0.1204AwayString$. Of course, the $HomeString$ and $AwayString$ variables are the number of home or away games that have been played in a row up to the current matchday, as defined above. Due to our restriction that there can be no more than two home or away games in a row, these variables can only take the values 0, 1, and 2. Both coefficients for the variables are positive, so we want to maximize these values as much as possible to increase advantage. Furthermore, notice that the HomeString variable has a higher weighting on it. This indicates that given the choice between scheduling two home games in a row or two away games in a row, we would want to choose home games over away games.

To illustrate, consider the possible home and away combinations if we played four games where two games are home and two are away. Clearly, we have $\binom{4}{2} = 6$ ways of doing this. The possible combinations are HHAA, AAHH, HAHA, AHAH, HAAH, and AHHA where A is away and H is home. Consider the first option: HHAA. The first game would be the first home game in a row, and zeroth away games in a row. The second game would be the second home game in a row and the zeroth away game in a row. The third game would be the zeroth home game in a row and the first away game in a row. Finally, the fourth home game would be the zeroth home game in a row and the second away game in a row. Therefore, the $HomeString$ vector is (1,2,0,0) and the $AwayString$ vector is (0,0,1,2). This means that the advantage corresponding to this schedule is 0.2652(3) + 0.1204(3) = 1.1568. The advantage for the AAHH schedule is the same (it is simply the opposite schedule of the first). The advantage for HAHA and AHAH are both 0.7712. The advantage for HAAH is

0.8916 and the advantage for AHHA is 1.0364. As illustrated by this example, it is clear that pairing the home and away games together increases a teams seasonal advantage. The same results are obtained on a larger scale.

We now return to the scheduling. As before, we begin with scheduling the 10^{th} placed team first. Since we want to maximize the number of home pairings and away pairings, we obtain a schedule that looks like this:

	Matchday								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Team 10	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2
Team 8	7	4	2	9	6	1	10	5	3
Team 7	8	2	6	10	1	5	9	3	4
Team 6	3	10	7	1	8	9	2	4	5
Team 5	4	9	1	2	10	7	3	8	6
Team 4	5	8	9	3	2	10	1	6	7
Team 3	6	1	10	4	9	2	5	7	8
Team 2	1	7	8	5	4	3	6	10	9
Team 1	2	3	5	6	7	8	4	9	10

home
away

After filling in the this teams home/away schedule, we must denote the status of the corresponding fixtures. Obviously, if the 10^{th} placed team plays an opponent at home on a given matchday, that opponent must be playing the 10^{th} placed team away on that same matchday. After accounting for this, we can move onto the 9th placed teams scheduling by again maximizing home pairing and away pairings. Obviously, we must keep in mind that we must maximize the number of home/away pairings while considering the match against team 10 is already set. The resulting schedule, again accounting for corresponding fixtures, looks like this:

	Matchday								
	1	2	3	4	5	6	7	8	9
Team 10	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2
Team 8	7	4	2	9	6	1	10	5	3
Team 7	8	2	6	10	1	5	9	3	4
Team 6	3	10	7	1	8	9	2	4	5
Team 5	4	9	1	2	10	7	3	8	6
Team 4	5	8	9	3	2	10	1	6	7
Team 3	6	1	10	4	9	2	5	7	8
Team 2	1	7	8	5	4	3	6	10	9
Team 1	2	3	5	6	7	8	4	9	10

home
away

While it may be tempting to move onto the next team and create their home/away schedule, we must address a potential problem. Had we moved on to the next team, we could create inconsistencies in the rules established due to the corresponding fixtures. For example, we may inadvertently force a team to play three away matches in a row due to the fact we already set the schedules for the teams above it. Therefore, after scheduling for the 9^{th} placed team, we must look through the rest of the schedule and mark which matches are now forced to be home and away. Take the instance where a team currently has the following home/away schedule: H, __, H. Because a team may not play three home matches in a row, the blank must be an away match (and thus the team they are playing must also be at home for that corresponding fixture). Another example would be __, A, A, __. Both blanks must be home matches due to the same rule (and again, the opponents which these blanks correspond to must play away for those fixtures). A final, yet vital consideration involves the mirroring technique. Since we are only scheduling for the first nine games, we must keep in mind that the home/away scheduling for the next nine games are the exact opposite; if a game for a team on matchday 1 is at home, then it must be away on matchday 10. The 2-game max consecutive home/away rule therefore prohibits, for example, a team starting their season with two home games with their ninth game away. Due to the mirroring, this would force games ten and eleven to both be away and thus create a string of three away games which is not allowed. Ironically, even after scheduling for the first two teams, the rules established forces the schedules to look like this:

	Matchday								
	1	2	3	4	5	6	7	8	9
Team 10	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2
Team 8	7	4	2	9	6	1	10	5	3
Team 7	8	2	6	10	1	5	9	3	4
Team 6	3	10	7	1	8	9	2	4	5
Team 5	4	9	1	2	10	7	3	8	6
Team 4	5	8	9	3	2	10	1	6	7
Team 3	6	1	10	4	9	2	5	7	8
Team 2	1	7	8	5	4	3	6	10	9
Team 1	2	3	5	6	7	8	4	9	10

home
away

After this we can move onto scheduling the blank spaces for the next lowest ranked available team (in this case team 8). Of course, we need to make sure we maintain our rule on sets of home and away games. The final schedule is as follows:

	Matchday																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Team 10	9	6	3	7	5	4	8	2	1	9	6	3	7	5	4	8	2	1
Team 9	10	5	4	8	3	6	7	1	2	10	5	4	8	3	6	7	1	2
Team 8	7	4	2	9	6	1	10	5	3	7	4	2	9	6	1	10	5	3
Team 7	8	2	6	10	1	5	9	3	4	8	2	6	10	1	5	9	3	4
Team 6	3	10	7	1	8	9	2	4	5	3	10	7	1	8	9	2	4	5
Team 5	4	9	1	2	10	7	3	8	6	4	9	1	2	10	7	3	8	6
Team 4	5	8	9	3	2	10	1	6	7	5	8	9	3	2	10	1	6	7
Team 3	6	1	10	4	9	2	5	7	8	6	1	10	4	9	2	5	7	8
Team 2	1	7	8	5	4	3	6	10	9	1	7	8	5	4	3	6	10	9
Team 1	2	3	5	6	7	8	4	9	10	2	3	5	6	7	8	4	9	10

home
away

Finished Scheduling

This is the resulting schedule of our formulation. Upon visual inspection, several interesting trends become quite apparent. First and most importantly, lower strength teams received preference in this scheduling formulation. As team strength increased, the strength of opponents varied less and less from game to game. This moved higher-ranked teams away from having a balanced schedule. This was intentional since they were given the least preference. It should also be noted that while the overall advantage for each team tends

to decline from the stronger teams to the weaker teams, it does so by small intervals. This indicates that teams are actually all receiving roughly equitable schedules, so there is not an unreasonable amount of advantage given to any one team that could lead to complaints of blatant unfairness.

Interestingly, all team schedules begin with a game against a team with similar rank, indicating that the formulation attempts to give the current team the easiest start to the season possible. On the opposite end, each team finishes the schedule playing the team with opposite rank (10 against 1, 9 against 2, and so on). So to a weaker team, it may appear that they will be stuck with a hard end to their season, while strong teams get an easy end. Though the formulation tells us that this scheduling is the most fair, individual teams may perceive this to be unfair. This is because a special emphasis is often placed at the end of season, when many teams look to increase their goal counts or try to make a final push to increase their league rank. Some might be unhappy with the psychological effect that this perception could incur. Accounting for this would likely lead to a more complicated model, one that would be outside the capabilities of our solver used.

Further Applications

Our formulation explored the best scheduling for a theoretical half-sized Premier League. With a more efficient calculation algorithm and/or more powerful solver, our formulation could easily be scaled up to the full 20-team, 38-fixture problem. In addition to the optimizations we already have, other factors could be added to the model, such as number of days of rest between games and distance traveled. These variables do not seem to have as much impact or significance as the ones presently considered although they are shown to contribute. If these factors were accounted for, they would be implemented in a very similar fashion to the methods we have already established.

Though our simulated league does not have them, a real-life scheduling of games could contain many restrictions that must be translated into constraints. For example, a certain city may not be able to accommodate a game on a certain date. Additional constraints could range from city mandates, team requests, or individual holidays. Nevertheless, our model offers a solid basis for producing a fair schedule with the techniques made available to us.