MIT MANAGEMENT SLOAN SCHOOL

Master of Business Analytics

15.093: Optimization

Optimal European Soccer Matches Scheduling

Andrea ZANON

Hamza ZERHOUNI

github.com/azanon00/Optimization_Project

December 9, 2022

Contents

1 Introduction

Footballers play many games per season, muscle and mind stress makes them injury-prone if rest periods are not carefully planned. Often, many coaches connect injuries with tight schedules and consider little rest as the reason why their team does not perform well. While not all injuries depend on not having enough rest, it would be extremely beneficial to optimize the games' schedule such that rest is maximized. This is of great interest not only for players themselves, but also for supporters, sponsors, and TV broadcasters, since teams would be allowed to perform at their best.

2 Data

External data needed for our project is the true real-life schedules of the leagues. This data is necessary to obtain a baseline to measure the performance of our model.

Other than that, no external data is needed. What instead is required is domain expertise and knowledge of the rules that a soccer schedule must follow.
For instance, it is important to notice that our project focuses on optimizing the schedule of one single league, thus knowing in which days of the week each league is allowed to play is fundamental. Also, it is necessary to know how many teams play in that league.

3 Parameters

- $i = 1, ...N$: number of teams in the league.
- $w = 1, ..., W$: number of weeks to schedule.
  In particular, we have that $W = N - 1$, meaning that if there are 20 teams, then we need to schedule 19 weeks. The reason is that each team plays one game per week against a different opponent until all teams played each other. Note that in general each team throughout the year plays 2(N-1) games since teams play each other twice in total (once in the home stadium of each), but the order is the same in the first and second part of the tournament, thus optimizing the first half is enough.
- $d = 1, ..., D$: days of the week
  We defined the formulation such that $d = 1$ corresponds to Tuesday, and $d = 7$ to Monday, since Monday is part of the previous soccer week.
- $R_d = \begin{cases} 1, & \text{if game can be played on day d} \\ 0, & \text{otherwise} \end{cases}$
  In our case, $R_d = 0$ for $d = 1, 2, 3$, and $R_d = 1$ for $d = 4, 5, 6, 7$ because league games are only played on Friday, Saturday, Sunday and Monday
- $C_{tt'} = \begin{cases} \frac{1}{t'-t}, & \text{if t' > t} \\ 0, & \text{otherwise} \end{cases}$
  The most important parameter in our formulation is $C \in R^{TxT}$, the cost matrix, where $T = 7W + D$ is the total number of days in our planning horizon.
  $C_{tt'}$ represents the cost of playing in day t, followed by day t'.
  The reason why the cost is modeled in this way, instead of simply considering

$C_{tt'} = \Delta t = t' - t$ is that evenly spaced rest periods are preferred. With a concrete example, we want to model that playing on days 1, 3 and 5 is better than playing on days 1, 2 and 5. We see that using our approach, the second sequence is worse than the first one, as it should be. Using the other approach instead, both sequences give the same cost.

Sequence	$\frac{1}{t'-t}$	$t' - t$
(1, 3, 5)	$\frac{1}{2} + \frac{1}{2} = 1$	$2 + 2 = 4$
(1, 2, 5)	$\frac{1}{1} + \frac{1}{3} = 1.33$	$1 + 3 = 4$

Table 1: Example of comparisons between the two approaches for cost time

4 Formulation

4.1 Decision variables

Our decision variable is always

$$x_{ijwd} = \begin{cases} 1, & \text{if team i plays team j on day d of week w} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

4.2 Objective value

Through our objective function, we want to maximize the rest between games, which is equivalent to minimizing the cost of having little rest between each pair of games.

$$\min_x \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{w=1}^{W-1}\sum_{d=1}^{D}\sum_{d'=1}^{D} C_{7w+d,7(w+1)+d'}\; x_{ijwd}\; x_{ik(w+1)d'} \tag{2}$$

The product of the two decision variables will be 1 when both are 1, meaning that team i is playing team j on day d of week w, and team k in day d' of the following week. And we are add the cost of resting just between these two days and not more. When either $X_{ijwd} = 0$ or $X_{ik(w+1)d'} = 0$, the corresponding cost is not considered.

One problem with formulating the objective value in this way is that, as we can see, it is non-linear. However, it can be linearized in the following way:

$$\min_x \sum_{i,j,k,w,d,d'} C_{7w+d,7(w+1)+d'}\; x_{ijwd}\; x_{ik(w+1)d'} \tag{3}$$

$$\Longleftrightarrow$$

$$\begin{aligned} \min_x \quad & \sum_{i,j,k,w,d,d'} C_{7w+d,7(w+1)+d'}\; a_{ijkwdd'} \\ \text{s.t.} \quad & a_{ijkwdd'} \leq x_{ijwd} \\ & a_{ijkwdd'} \leq x_{ik(w+1)d'} \\ & a_{ijkwdd'} \geq x_{ijwd} + x_{ik(w+1)d'} - 1 \end{aligned} \tag{4}$$

4.3 Constraints

- One first constraint is that each team can never play against itself. There are many ways to represent this, but in integer optimization it is convenient to have many different constraints (see *"modeling with exponential number of constraints"* in the lecture) to get as close as possible to the ideal formulation. Therefore,

$$X_{iiwd} = 0 \quad \forall i = 1, ..., N,\ w = 1, ..., W,\ d = 1, ..., D$$

- Of course, it is intuitive that if team A plays team B, then team B plays team A, meaning that by fixing w and d, the slice of the tensor X_{ijwd} obtained is symmetric. We can model it in the following way:

$$X_{ijwd} = X_{jiwd} \quad \forall i = 1, ..., N,\ j = 1, ..., N,\ w = 1, ..., W,\ d = 1, ...D$$

- Each team plays at most one opponent per week. Remember that the first day of the week is considered to be Tuesday, so Monday in our formulation is included in the previous week.

$$\textstyle\sum_{j=1}^{N} \sum_{d=1}^{D} x_{ijwd} \leq 1 \quad \forall i = 1, ..., N,\ w = 1, ..., W$$

- Another important constraint is in the total number of games played by each team. As we previously mentioned in the parameters section, each team plays all the other teams once, for a total of $N-1$ games.
  Notice that throughout a championship each team plays $2(N-1)$ games, but since the second half has the same order as the first, we just optimize over $N-1$ games. Combined the previous constraint, it is the same as modeling that teams play one and only one game per week. However, we chose to keep this formulation because it is more flexible, meaning that if we wanted to model that there can be some "break weeks" (due to international competitions or other events), we could simply change the right-hand side and keep the rest of the formulation unchanged.

$$\textstyle\sum_{j=1}^{N} \sum_{w=1}^{W} \sum_{d=1}^{D} x_{ijwd} = N-1 \quad \forall i = 1, ..., N$$

- Imposing the total number of games is not sufficient, we also need to model that teams cannot compete against each other more than once.

$$\textstyle\sum_{w=1}^{W} \sum_{d=1}^{D} x_{ijwd} \leq 1 \quad \forall i = 1, ..., N,\ j = 1, ..., N$$

- Furthermore, as we said the national leagues play over the weekend, therefore there are no games on Tuesday, Wednesday and Thursday.

$$x_{ijwd} \leq R_d \quad \forall i = 1, ..., N,\ j = 1, ..., N,\ w = 1, ..., W,\ d = 1, ..., D$$

5 Different approaches

To tackle our problem, we tried a number of different approaches and then selected the best one.

5.1 Optimize the entire league

The first approach we tried was simply to translate the above linear optimization formulation in Julia, letting Gurobi solve it.

While the solver efficiently returns the solution for a limited number of teams, we quickly noticed that the problem does not scale, due to the huge number of variables ($\approx\ D \cdot N^3$) and constraints ($\approx\ 2 \cdot N^3 + 9 \cdot N^2 + N$). Since the problem must be solved for N = 20, coming up with clever heuristics was a necessity.

We let this optimal approach run overnight to get a reference, but still it did not converge. We therefore put a 6h time limit in order to have an approximate solution to the optimal one.

5.2 Split teams in groups

One first approach to solve this issue was to split the 20 teams between 4 groups of 5 teams each. Every period of 5 weeks, each team of group p_i $(i = 1, 2, 3, 4)$ plays only against group p_j $(j \neq i)$, and in the following 5 weeks against another group; this happens from week 1 to week 15. Between week 16 and 19, there are intra-group games.
While this approach scales, it cannot be used to solve our problem since having an odd number of teams in each group means that in the intra-group phase there would be one different team per week that does not play.
This approach would only work if and only if we can split the number of teams into an even number of groups, each group having an even number of teams.

5.3 Greedy approach (optimize P weeks at a time and freeze them)

A better approach is to optimize P weeks at a time for all the 20 teams, freeze that period and then move to the following P weeks. The solver is called iteratively (a total of $\lceil \frac{W}{P} \rceil$ times) until all the games are scheduled. Each iteration optimizes the period between W_{in} and W_{end}. In our project, we chose $P = 4$. We define as W_{in} the initial week of the interval being consider, and W_{end} the final one; $W_{end} = W_{in} + P - 1$.
While the formulation is mostly the same, we need to make some changes to allow the solver to have "memory"; in other words, to consider as fixed the matches scheduled in the previous iterations.

To do so, we add the variable $Y \in R^{N\,x\,N}$:

$$Y_{ij} = \begin{cases} 1, & \text{if team i already played against team j} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

- Y^{old}: memory of the previous iterations, meaning that after very every iteration, we update Y^{old} with the solution of Y. Initially, $Y^{old} = 0$

- During the iteration, Y is updated: the change reflects the games that are scheduled while optimizing the P weeks considered. The following constraints need to be added:
  - If team i played team j in the past, they cannot be matched again.
    $$\sum_{w=W_{in}}^{W_{end}} \sum_{d=1}^{D} x_{ijwd} \leq 1 - Y_{ij}^{\text{old}} \quad \forall i = 1, ..., N,\ j = 1, ..., N$$
  - If team i is assigned to play against team j, we update Y_{ij}
    $$Y_{ij} \geq \sum_{w=W_{in}}^{W_{end}} \sum_{d=1}^{D} x_{ijwd} \quad \forall i = 1, ..., N,\ j = 1, ..., N$$
  - The matrix Y is symmetric, because if team i played team j then the opposite is also true.
    $$Y_{ij} = Y_{ji} \quad \forall i = 1, ..., N,\ j = 1, ..., N$$

We must pay attention to the fact that the **objective function** changes as well. In fact, by solving the formulation with the old objective function, the solver would put $Y_{ij} = 1, \quad \forall i = 1, ..., N,\ j = 1, ..., N$ at each iteration.
To solve this issue, we simply add a penalty term in the objective

$$\min_{x} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{w=1}^{W-1} \sum_{d=1}^{D} \sum_{d'=1}^{D} C_{7w+d,7(w+1)+d'}\, X_{ijwd}\, X_{ik(w+1)d'} + \sum_{i=1}^{N} \sum_{j=1}^{N} Y_{ij} \tag{6}$$

Thus, after every iteration, we update Y^{old} as follows: $Y^{\text{old}} \mathrel{+}= value.(Y)$ and we are ready for the following iteration. As we said, we stop when we reach the last period (W).
Solving each iteration requires approximately 10 minutes, thus the whole solution is found in 50 minutes, a big improvement compared to the unscalable optimal approach.

5.4 Greedy approach with rolling window (optimize P weeks at a time and freeze only the first)

This second greedy approach is similar in spirit, but the execution is different. Instead of freezing the entire period, we fix just the first week at each iteration.

While constraints and objective function are exactly the same as the previous greedy approach, there is a difference in how the matrix Y is updated. For each period, only the first week is updated, except in the last period where the whole matrix is.

Algorithm 1 Update Y

```
if last period == TRUE then
    $Y^{\text{old}} \mathrel{+}= value.(Y)$
else
    for i = 1,...,N; j = 1,...,N; d = 1,...,D do
        if $X_{ijW_{in}d} == 1$ then
            $Y_{ij}^{\text{old}} \mathrel{+}= Y_{ij}$
        end if
    end for
end if
```

Again, solving each iteration requires approximately 10 minutes; however, here we need to iterate 16 times, thus the solution is found in 2h45 minutes.

5.5 Greedy approaches with random restart

After we use the previously described heuristics to find the solution we can implement a random restart algorithm in order to further improve their quality.

The idea is the following: the number of weeks to re-optimize is randomly chosen within 2 and 5 (cannot be too big otherwise the algorithm does not scale), and the initial week is randomly chosen as well.
We get all the teams that played against each other in that period, and we reset $Y_{ij}^{\text{old}} = 0$ if team i and team j played against each other in the period we are re-optimizing.
At this point, the solver is called and matches are rescheduled. We compare the new objective function with the old one; if there is an improvement, we update the calendar, otherwise the previous one is kept.
This whole process is repeated K times.

The time required to solve the problem increases and is equal to:

$$T = (\text{time to solve the heuristic}) + K \times (\text{average time of re-optimizing the period})$$

Considering that the average period encompasses 3 weeks, assuming that we make 10 random restarts the whole process takes $\approx$ 1h30 minutes for the approach (5.3) and $\approx$ 3h20 minutes for approach (5.4).

6 Results

6.1 How to evaluate results

First, all our methods are evaluated compared to the real *Serie A* schedule. This is taken as a baseline. We expect optimization to have an edge, but not to be huge since in both real life and our formulation there is one game per week.
Then, all the models are compared against each other and the best will be chosen.

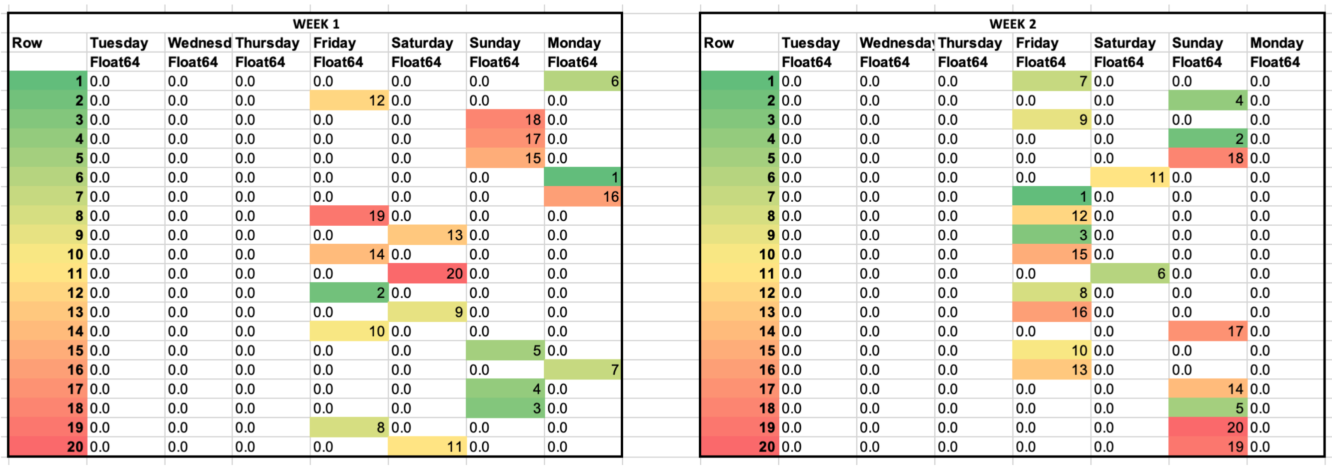
Since in the heuristics there is $\sum Y_{ij}$ in the objective, note that in order to make all the values comparable, a new function is coded that given X returns the value of the objective function.

6.2 Our results

Models	Objective function	Time (s)
Baseline (Real life)	55.76	/
Freeze entire optimized period (5.3)	56.95	3,084
Freeze first week of optimized period (5.4)	51.68	9,881
(5.3) + Random restart	56.20	5,373
(5.4) + Random restart	51.68	11,932
Optimal model (time limit)	51.46	21,600

Clearly, the optimal solution would be one that presents a good trade-off between performance and computation time. The optimal model gives us the best performance as it is the model that maximizes the most the rest time between games, but it is computationally heavy and not scalable.
By looking at this table we see that even though freezing 4 weeks entirely is much faster than freezing just the first week of the period, it does not make sense to implement it since there is no improvement over the baseline. We also observe that random restarts helps improve approach **(5.3)**, but have no impact on **(5.4)**, meaning that probably we are pretty close to an optimal solution.
All things considered, the best heuristic approach is ***Freeze first week of period*(5.4)**, which leads to the best solution among all corresponding to a **7.3%** improvement over the baseline. Besides, this approach is very satisfactory as the solution is very close to the solving the optimal problem for 6h, with only a gap of **0.3 %** in **54.25 %** less time.

The calendar returned by that approach can be easily visualized. Here, for ease of understanding (to avoid displaying all the 19 weeks), we show the scheduling obtained for the first two weeks.



WEEK 1							
Row	Tuesday	Wednesd	Thursday	Friday	Saturday	Sunday	Monday
	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	0.0	0.0	0.0	0.0	0.0	0.0	6
2	0.0	0.0	0.0	12	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	18	0.0
4	0.0	0.0	0.0	0.0	0.0	17	0.0
5	0.0	0.0	0.0	0.0	0.0	15	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	1
7	0.0	0.0	0.0	0.0	0.0	0.0	16
8	0.0	0.0	0.0	19	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	13	0.0	0.0
10	0.0	0.0	0.0	14	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	20	0.0	0.0
12	0.0	0.0	0.0	2	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	9	0.0	0.0
14	0.0	0.0	0.0	10	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	5	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	7
17	0.0	0.0	0.0	0.0	0.0	4	0.0
18	0.0	0.0	0.0	0.0	0.0	3	0.0
19	0.0	0.0	0.0	8	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	11	0.0	0.0

WEEK 2							
Row	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Monday
	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	0.0	0.0	0.0	7	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	4	0.0
3	0.0	0.0	0.0	9	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	2	0.0
5	0.0	0.0	0.0	0.0	0.0	18	0.0
6	0.0	0.0	0.0	0.0	11	0.0	0.0
7	0.0	0.0	0.0	1	0.0	0.0	0.0
8	0.0	0.0	0.0	12	0.0	0.0	0.0
9	0.0	0.0	0.0	3	0.0	0.0	0.0
10	0.0	0.0	0.0	15	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	6	0.0	0.0
12	0.0	0.0	0.0	8	0.0	0.0	0.0
13	0.0	0.0	0.0	16	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	17	0.0
15	0.0	0.0	0.0	10	0.0	0.0	0.0
16	0.0	0.0	0.0	13	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	14	0.0
18	0.0	0.0	0.0	0.0	0.0	5	0.0
19	0.0	0.0	0.0	0.0	0.0	20	0.0
20	0.0	0.0	0.0	0.0	0.0	19	0.0

Figure 1: Scheduled for the first two weeks returned by approach (5.4)

7 Conclusion and possible improvements

To conclude, with clever heuristics, we are able to make the model scale. Of course, all of them still take some time, but it is important to consider the notion of *practability*, meaning that the problem can be solved in times and sizes appropriate for the application. Scheduling soccer games usually happens once at the beginning of the tournament, and while it is important to have a problem that scales: it must run quick to accommodate potential last minute requests, but runtimes in the order of a couple of hours is acceptable. Overall, we can say we have a solution that in a feasible time substantially improves upon the baseline and is closed enough to the optimal solution to be considered as efficient.

There also are some potential improvements to our formulation. In particular, considering that some teams play other competitions as well (Champions League, Europa League, ...), which could be done by adding one constraint once we know which teams take part in that competition. Note that one complication is that in these competitions we do not know who will go at the knockout stage, but we could model that using stochastic optimization. In that case, the here-and-now decision is when to schedule league games, while the wait-and-see decision is when to schedule the other competition games; note that it is scenario dependent since in each scenario the team reaches a certain stage of the competition.

In addition, it could be possible to model the additional fatigue of traveling long distances between one game and the other, considering the time-zone impact as well (relevant when international competitions are also included in the modeling).

Moreover, one possible improvement would be to formulate the problem at the player level instead of team level. The reason why this would be useful is that many players play for their national team in addition to competing with their club. However, this adds many layers of complexity. First of all, the number of variables and constraints would be incredibly higher; also, games are of course scheduled between teams, not between players, thus we would need to keep track to the team(s) each player is assigned to.