Modelling & solving the Sports Tournament Scheduling (STS) problem

Leonardo Billi leonardo.billi@studio.unibo.it,
Leonardo Massaro leonardo.massaro2@studio.unibo.it,
Giorgio Scavello giorgio.scavello@studio.unibo.it

September 6, 2025

1 Introduction

The aim of this project is to resolve a particular instance of the Sports Tournament Scheduling problem. Even though this type of task has been thoroughly analyzed in the literature, it was not easy to find models which could be used for all the three approaches which are described in this paper. A great inspiration was given by the paper of Kendall et al. (2010) where a variety of approaches is exposed which helped also in understanding quickly the state of the art. Starting from this then it was possible to read other connected papers which were published later such as the work of Larson et al. (2014) which helped in formalizing the SMT model.
The models used in the three approaches do not share many similarities, this was chosen so that it was possible to exploit the strengths of each of them independently. It was however a common idea that using a preprocessing method would allow us to already have defined matches for a given week and inserting them in the schedule respecting the period constraint. This method is for sure the circle method which allows to mathematically compute all the matches which are played in Single Round Robin Tournament, as in our case.

The following description takes inspiration from the work of Lambrechts et al. (2018): imagine having n teams where n is even. Then team 1 is fixed at the center of the circle while the remaining teams are placed on the border of it. The team at the top position of the circle is matched with the team at the center. After this we match the first team we find clockwise to the first team counterclockwise. When all the teams are matched then the teams on the circle rotate clockwise and the process is repeated until the teams on the circle are on the initial position.
This method allows to satisfy two of the main constraints which were necessary to find a valid solution, as a schedule obtained via the circle method will have each team play against a different one each week and also will not allow a team to play more than once a week. The constraint about period and also other constraints which are necessary to obtain a satisfying result are encoded in different ways in the three different approaches.
It is necessary to define the parameters which are common to all sections:

- N is the number of teams and it is assumed to be even.
- W is the number of weeks and the number of games each team plays in total. By definition, $W = N - 1$.
- P is the number of periods in each week, therefore the number of games in each week. By definition, $P = \frac{N}{2}$.

All the experiments for each approach were conducted on a machine with an AMD Ryzen 7 8-core processor running at 4.2 GHz. The computational environment was a Docker container using a slim Debian Linux distribution, and each experimental run was limited to 300 seconds as stated in the problem specs.

2 CP Model

The CP approach was initially developed using a basic and naive formulation, in order to provide a baseline model against which subsequent implementations could be evaluated. Later models demonstrate how the formulation can be improved through a more efficient encoding of the decision variables and constraints, as well as through the introduction of symmetry-breaking constraints and preprocessing phases.

As presented in Kendall et al. (2010), the optimization problem can be divided into two parts (first scheduling, then breaking), thereby reducing the complexity of a single large search space.

The models for generating a calendar without considering home/away patterns are defined as follows:

- **Naive Model**: basic encoding
- **Constraint v1 Model**: improved encoding of the decision variables
- **Constraint v2 Model**: inclusion of symmetry-breaking constraints
- **Round Robin Model**: addition of a round-robin preprocessing stage to significantly reduce the search tree

Any valid solution generated by the above models can be used as input for the following:

- **HAP Model**: takes a valid calendar and assigns a balanced home/away pattern

2.1 Decision Variables

- **Naive Model**: decision variables are encoded as follows: For each period $p \in \{1, \dots, \frac{n}{2}\}$ and each week $w \in \{1, \dots, n-1\}$, we define two variables:

$$\text{home\_team}_{p,w} \in \{1, \dots, n\}, \quad \text{away\_team}_{p,w} \in \{1, \dots, n\},$$

  which indicate respectively the team playing at home and the team playing away in period p of week w. Each match is therefore represented by the pair $\big(\text{home\_team}_{p,w}, \text{away\_team}_{p,w}\big)$, and the schedule is fully determined by these values.

- **Constraint v1 and v2 Models**: decision variables are encoded in a more efficient way: represented by the array

$$\text{calendar} \in \{1, \dots, \text{max\_m}\}^{\text{periods} \times \text{weeks}},$$

  where $\text{max\_m} = \frac{n^2}{2} - \frac{n}{2}$ is the maximum number of possible matches between n teams. Each entry $\text{calendar}_{p,w}$ is a decision variable taking values in the index set MATCHES $= \{1, \dots, \text{max\_m}\}$, which refers to a unique pair of teams stored in the precalculated array `matches`. Thus, $\text{calendar}_{p,w}$ encodes the match scheduled in period $p \in \{1, \dots, n/2\}$ and week $w \in \{1, \dots, n-1\}$, and the complete tournament schedule is determined by the assignment of all these decision variables.

- **Round Robin Model**: In this model, the primary decision variable is the array

$$\text{permutation\_calendar} \in \{1, \dots, \text{periods}\}^{\text{periods} \times \text{weeks}},$$

  where each entry $\text{permutation\_calendar}_{p,w}$ represents a permutation index that selects which match from the preprocessed `calendar` array is scheduled in period $p \in 1..\text{periods}$ during week $w \in 1..\text{weeks}$. By assigning values to all entries of permutation_calendar, the model determines the full tournament schedule, since each index corresponds to a specific pair of teams stored in the parameter array `calendar`.

- **HAP Model**: In this model, the main decision variables are represented by the array

$$\text{home\_team\_index} \in \{1, 2\}^{\text{periods} \times \text{weeks}},$$

  where each entry $\text{home\_team\_index}_{p,w}$ indicates which team in the match scheduled at period $p \in 1..\text{periods}$ and week $w \in 1..\text{weeks}$ plays at home. The matches themselves are fixed and provided by the parameter array $\text{sol}[p, w, 1..2]$.

2.2 Objective Function

The chosen objective function evaluates the home/away balance by considering only the number of home matches played by each team.

$$\min\ \text{objective\_function} = \max_{t \in \text{TEAMS}}(\text{home\_games}_t) - \min_{t \in \text{TEAMS}}(\text{home\_games}_t),$$

where

$$\text{home\_games}_t = \sum_{p=1}^{\text{periods}} \sum_{w=1}^{\text{weeks}} \mathbf{1}_{\{\text{sol}[p,w,\text{home\_team\_index}_{p,w}]=t\}},$$

and $\mathbf{1}_{\{\cdot\}}$ is the indicator function, equal to 1 if the condition is true and 0 otherwise. This objective minimizes the difference between the maximum and minimum number of home games assigned to any team.

Since each team plays an odd number of matches $(n-1)$, the minimum possible value of the function is 1. Thus, when the difference between the maximum and minimum values in the home-count array equals 1, we know that every team has achieved a balanced home/away count.

2.3 Constraints

In the initial model, all constraints were explicitly stated according to the problem description:

1. A single match must involve two distinct teams: encoded as

$$\text{home\_team}_{p,w} \neq \text{away\_team}_{p,w} \quad \forall p \in 1..\tfrac{n}{2},\ w \in 1..n-1.$$

2. Each team must play exactly once per week: encoded with the global `alldifferent` constraint:

$$\texttt{alldifferent}\big(\{\text{home\_team}_{p,w}, \text{away\_team}_{p,w} \mid p \in 1..\tfrac{n}{2}\}\big), \quad \forall w \in 1..n-1.$$

3. Each pair of teams must meet exactly once during the tournament: encoded as

$$\sum_{p,w} \big[(\text{home\_team}_{p,w} == i \land \text{away\_team}_{p,w} == j) \lor (\text{home\_team}_{p,w} == j \land \text{away\_team}_{p,w} == i)\big] = 1.$$

4. Each team can play at most twice in a single period across all weeks: encoded as

$$\sum_{w=1}^{n-1} \big[(\text{home\_team}_{p,w} == t) \lor (\text{away\_team}_{p,w} == t)\big] \leq 2, \quad \forall t \in \{1, \dots, n\},\ p \in 1..\tfrac{n}{2}.$$

Starting from the **Constraint v1 Model**, some constraints were encoded more efficiently. Thanks to the preprocessing phase that enumerated all possible matches in advance, the first constraint (distinct teams per match) became redundant and was therefore removed. The remaining constraints were reformulated as follows:

(3) Each pair of teams must meet exactly once during the tournament: encoded as

$$\texttt{alldifferent}\Big(\{\text{calendar}_{p,w} \ \mid\ p \in 1..\text{periods},\ w \in 1..\text{weeks}\}\Big).$$

where 'calendar' is an index pointing to a valid match from the preprocessed array of matches.

Symmetry-Breaking Constraints. Since our objective is to find at least one feasible solution, we can exploit symmetries in the validity of a calendar to reduce the search space and hopefully obtain faster solutions. Beginning with the **Constraint v2 Model**, we introduced the following symmetry-breaking constraints:

1. **Breaking the symmetry due to team exchange in a single match.** In the representation of matches, each game is defined by a pair of teams. Exchanging the order of these two teams yields an equivalent match (team i vs. team j equals team j vs. team i). By fixing the representation of each match to a canonical form—for instance, always storing the smaller team index first—we eliminate this redundant symmetry. This reduces the number of match encodings by half, directly shrinking the branching factor in the search.

2. **Breaking the symmetry due to permutations of weeks.** Without additional constraints, any feasible schedule can be cyclically permuted across the weeks, yielding $n-1$ equivalent calendars. To avoid exploring these symmetric variants, we impose a lexicographic ordering on the weekly assignments (`lex_chain(calendar)` in the model). This ensures that only one canonical representative among these permutations is considered during the search, which can cut down the solution space by a factor of up to $(n-1)!$.

3. **Fixing a starting point for the calendar.** To further reduce symmetry, we fix the very first match of the tournament, e.g., `calendar[1,1] == 1`, which anchors the schedule to a unique starting configuration. This breaks the symmetry of arbitrary rotations or reorderings of matches in the first period, helping the solver converge more quickly by avoiding equivalent calendars that differ only in their initial match choice.

2.4 Validation

All models were implemented in MiniZinc 2.9.3 and tested with the bundled solvers `Gecode` and `Chuffed`.

As required by the problem, a time limit of 300 seconds was imposed, including preprocessing as well as both the satisfiability and optimization phases (when applicable).

Experimental Results. In the following table we present the experimental results for each 'n' for a selection of interesting CP approaches. The column names are composed in this way: the first word indicates which model was used between Round Robin Model, Constraint V2 model, Constraint V1 model, Naive Model; the second word indicates the solver between Chuffed and Gecode; the last word 'opt', if present, indicates that the model tried to optimize the home/away balance, so the results are the values of the objective function.

Table 1: Experimental results for different values of n. In columns with 'opt' suffix the result is the value of the obj function. In the other columns the value is the time in seconds

n	**RR chuf opt**	**CV2 chuf opt**	**CV2 gec**	**CV1 gec**	**Naive gec**
6	1	1	0	0	0
8	1	1	0	0	0
10	1	1	0	11	—
12	1	1	23	3	—
14	1	2	—	—	—
16	3	—	—	—	—

3 SMT Model

Starting from a very simple approach which aimed at imposing the constraints on an array, which would later be the solution, two different models were developed inspired by the work of Larson et al. (2014). Even though the model of this paper is CP based and it has a different goal, the adaptation to this framework gave satisfying results. In the following subsections, the characteristics of the two models employed will be presented, highlighting similarities and differences.

3.1 Decision Variables

The two models have similar decision variables as the work that inspired them is the same. However, as in the second model, there is a preprocessing phase which uses the circle method, then the Opp variables are defined just in the first model. Both models use Quantifier Free Linear Integer Arithmetic theory.
Here are the variables which were defined:

- $\text{Opp}_{ij} = k$ for $i = 0, \cdots, N-1$ and $j = 0, \cdots, W-1$ is a series of sort Int variables, which defines that team i+1 will meet team k in week j+1. The domain for this variable is $[1, \cdots, N]$. Obviously $\text{Opp}_{ij} \neq i+1$ as a team cannot play against itself.
- $\text{Per}_{ij} = k$ $i = 0, \cdots, N-1$ and $j = 0, \cdots, W-1$ a series of sort Int variables, which define that team i+1 in week j+1 will play in period k. The domain for this variable is $[1, \cdots, P]$.
- Home_{ij} for $i = 0, \cdots, N-1$ and $j = 0, \cdots, N-2$ is a series of sort Bool variables, which is true when the team i+1 is playing at home in week j+1 and it is false when the same team is playing away.

It could be possible to object that Per_{ij} is implied logically by Opp_{ij}, however this was kept in the model as it makes modeling simpler as it will be shown in section 3.3.
Even though 1-based indexing could be more understandable, 0-based indexing was preferred as it make easier to translate the SMT-LIB solution into the final solution.

3.2 Objective function

The function is defined as the sum of absolute differences between the home and away games of each team. This can be defined using the following indicator function on Home variables:

$$g_{ij} = \begin{cases} 1, & \text{if } \text{Home}_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

then the number of home games for each team will be obtained as $hg_i = \sum_{j=0}^{W-1} g_{ij}$. Having to minimize the difference between the home and away games, it is possible to define an objective function as:

$$obj = \sum_{i=0}^{N-1} |2hg_i - W| \tag{3.2}$$

The lower bound of this function is N as the minimum difference between home and away games is 1 for each team. The upper bound is obtained when half of the teams plays all the games at home and the remaining teams play all the games away, obtaining $obj = N \cdot W = N^2 - N$.

3.3 Constraints

3.3.1 Main problem constraints

The two models have constraints which are defined in a similar way. The following are the one which were explicitly defined:

- One constraint of the problem says every team plays with every other team only once. This is equivalent to say that all the opponents must be different, because the number of matches for each team must be equal to the number of possible opponents:

$$\bigwedge_{i=0}^{N-1} \bigwedge_{0 \le j < k < W} \text{Opp}_{ij} \neq \text{Opp}_{ik} \tag{3.3}$$

  This encoding was used in the first model, while in the second model it was not employed as the Opp_{ij} variables are not present and this constraint is respected by the circle method by construction.

- The second constraint says that every team must play once a week. Following the previous coding then it would be possible to define this as:

$$\bigwedge_{j=0}^{W-1} \bigwedge_{0 \le i < k < N} \text{Opp}_{ij} \neq \text{Opp}_{kj} \tag{3.4}$$

  However, coding two 'global' differences out of three constraints was shown to significantly reduce the performance of the model. For this reason, it was considered to exploit the variables Per_{ij}. This imposes that each team must have exactly one period then, after creating the following indicator function:

$$q_{ij}^{p} = \begin{cases} 1, & \text{if } \text{Per}_{ij} = p \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

  Then imposing the following constraint ensures that exactly two teams are assigned the same period each week:

$$\bigwedge_{j=0}^{W-1} \bigwedge_{p=1}^{P} (\sum_{i=0}^{N-1} q_{ij}^{p} = 2) \tag{3.6}$$

  Combining this to the fact a team can be assigned just one period by definition then a team plays exactly one match a week. This constraint is necessary for the first model, while in the second it is implied by the circle method.

- Another constraint says that every team plays at most twice in the same period over the tournament. The function at 3.5 is used also here to encode the following

$$\bigwedge_{i=0}^{N-1} \bigwedge_{p=1}^{P} (\sum_{j=0}^{W-1} q_{ij}^{p} \le 2) \tag{3.7}$$

  This constraint is present in both the first and the second model with the same formulation.

There are then structural constraints which must be imposed in order to obtain a correct solution, they are the following:

- When two teams are against each other then they should have the same period, in the first model they are encoded as follows:
$$\bigwedge_{0\le i\neq k<N} \bigwedge_{j=0}^{W-1} ((\text{Opp}_{ij} = k+1) \iff (\text{Opp}_{kj} = i+1 \ \land \ (\text{Per}_{ij} = \text{Per}_{kj}))) \tag{3.8}$$
It is necessary to say that it is actually constrained as two different implications as this improves significantly the performance.
Even though the concept is the same it is expressed slightly differently in the second model, here it is said that each team should have the same period as the team it is scheduled in that week by the circle method and a period different from all the other teams. Having defined opp_{ij}, as the opponent given of the $i+1$ team at week j (this is not a decision variable as it is already predefined by the circle method). Then being $k = \text{opp}_{ij} - 1$ the constraint is encoded as:
$$\bigwedge_{j=0}^{W-1} \bigwedge_{i=0}^{N-1} \text{Per}_{ij} = \text{Per}_{kj} \tag{3.9}$$
and also:
$$\bigwedge_{j=0}^{W-1} \bigwedge_{0\le i\neq l\neq k<N} \text{Per}_{ij} \neq \text{Per}_{lj} \tag{3.10}$$
- When two team play against each other then one should play at home and the other away, this is encoded differently in the models. In the first one it is encoded as:
$$\bigwedge_{0\le i\neq k<N} \bigwedge_{j=0}^{W-1} ((\text{Opp}_{ij} = k+1) \implies (\text{Home}_{ij} \oplus \text{Home}_{kj})) \tag{3.11}$$
In the second model, instead, as the circle method defines a match between i and k, then:
$$\bigwedge_{j=0}^{W-1} (\text{Home}_{ij} \oplus \text{Home}_{kj}) \tag{3.12}$$

3.3.2 Implied constraint

The only implied constraint across the two models is in the second model. Here the constraint defined at 3.6 was kept even though it was not necessary because the fact that a team does not play more than once in the week is implied by construction by the circle method. However, this greatly reduces the search space as the solver may need to deduce this relationship from 3.9 and 3.10.

3.3.3 Symmetry breaking constraints

The two models have similar symmetry breaking constraints:

- Break the home away flip by fixing the first team at home, meaning that Home_{00} should always hold. This is present in both models.
- Break the period symmetry by fixing the periods of the first week. This is done by imposing the following constraint:
$$\bigwedge_{p=1}^{P} (\text{Per}_{(p-1)0} = p \ \land \text{Per}_{(N-p)0} = p) \tag{3.13}$$
This is done similarly in the second model by using the matches which the circle method defines for the first week.
- Break the symmetry of the opponents in the first model by fixing two constraints: the first one is similar to the first symmetry breaking constraint, as Opp_{00} = N is fixed; the second constraint consists in fixing the opponents of the first team in descending order across weeks. This is encoded as follows:
$$\bigwedge_{j=0}^{W-2} \text{Opp}_{0j} > \text{Opp}_{0(j+1)} \tag{3.14}$$
This symmetry is not present in the second model as the opponents are already fixed by the circle method.

3.4 Validation

3.4.1 Experimental design

The models were written using the Python package Z3py for simplicity. From this, they were later converted to SMT-LIB by Barrett et al. (2016) using the function `.to_smt2()`. This allowed to compare the performance of three different solvers (cvc5, z3 and optimathsat) without necessitating a change in the code.
Even though some SMT solvers (i.e. Z3 and OptiMathSAT) include a minimization method which can be employed in SMT-LIB, this does not exist natively for all solvers. As modifying the code, even in a small part, depending on the solver used would make using solver independent language less useful then an offline optimization method via an iterative search was defined.
Here the solver is first asked to find a solution without taking into account the objective function, after this, if the solution is not optimal, it enters a loop where an upper bound on the objective function is imposed. This is upper bound is simply the objective function calculated at the previous iteration. A fixed lower bound saying that the objective function should be always greater or equal than N was used.
To make optimization faster it was also imposed that the number of home games of each team at each iteration, hg_{ik}, should be between two bounds calculated at the previous iteration:

$$\bigwedge_{i=0}^{N-1} (min_i \ hg_{i(k-1)} \leq hg_{ik} \leq max_i \ hg_{i(k-1)}) \tag{3.15}$$

This is imposed as half the teams should play $\frac{W-1}{2}$ home games and the remaining ones should play $\frac{W+1}{2}$ home games to achieve the optimal schedule.
Choices on the solvers were made so that they runs were more deterministic. For this reason it was chosen `--decision=internal` for cvc5 and `smt.phase_selection=4` for z3. In OptiMathSAT this cannot be set straight from the terminal so they were inserted in all the SMT-LIB file generated. Finally, different seeds were used and the obtained results are all similar to the one presented in 2, which were obtained using seed equal to 0.

3.4.2 Experimental results

In the following table, there are the values of the objective function as a function of n and the 6 different approaches. It is possible to see that Z3 is the most solid solver as it reaches solutions for N=18, finding an optimized one for the model using the offline method. While cvc5 stops finding solutions within the time limit after finding a solution for N=14, OptiMathSAT is able to find optimized solution until N=16 and goes very close for N=18. However, this is only for the model using the circle method, as for the other model the highest optimized solution is for N=8. This probably happens as the circle method prunes a lot of possible solutions.

n	**Z3 chan**	**Z3 pre**	**CVC5 chan**	**CVC5 pre**	**OptimathSAT chan**	**OptiMathSAT pre**
4	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
6	**6**	**6**	**6**	**6**	**6**	**6**
8	**8**	**8**	**8**	**8**	**8**	**8**
10	**10**	**10**	**10**	**10**	32	**10**
12	**12**	**12**	**12**	**12**	18	**12**
14	28	**14**	30	**14**	-	**14**
16	22	**16**	-	-	-	**16**
18	60	**18**	-	-	-	20
20	-	-	-	-	-	-

Table 2: The 6 versions contain the name of the solver used and the name of the model employed. Chan is for the first model, pre is for the model with the preprocessing.

4 MIP Model

The implementation supports four variable encodings (referred to later as `base`, `i!=j`, `i<j`, `prepro`) and an optional optimization objective to minimize home/away imbalance.

4.1 Decision variables

Full oriented variables (`base/v1`) Binary variables

$$x_{w,p,i,j} \in \{0,1\}$$

for all $w \in W$, $p \in P$, $i \in T$, $j \in T$. Interpretation: $x_{w,p,i,j} = 1$ means in week w, period p, team i plays team j (orientation matters: i at home, j away).

Oriented without self loops (`i!=j/v2`) Binary $x_{w,p,i,j}$ only for $i \neq j$ (removes trivial $i = j$ variables).

Unordered pairs (`i<j/v3`) Binary variables only for unordered pairs:

$$x_{w,p,i,j} \in \{0,1\}, \qquad \text{for } i < j,$$

meaning the unordered pair (i,j) is scheduled at (w,p); orientation is not encoded in x. When the optimization requires orientation (balanced objective), additional binary orientation variables are introduced, for each variable x:

$$h_{w,p,i,j} \in \{0,1\} \quad \text{for } i < j.$$

Preprocessed/v4 (`circle method`) As said in the introduction we use the circle method to generate a possible solution (I also added a randomize to swap home-away). So instead of deciding the full oriented assignment $x_{w,p,i,j}$, we fix per-week pairs and only decide a permutation assigning each precomputed pair to a period, reducing the number of assignment variables needed.

$y_{w,k,p} \in \{0,1\}$ (Binary), for every week w, precomputed pair index $k \in \{0,\dots,p-1\}$ and period $p \in P$.

When the optimization requires orientation, the additional variable is $h_{w,k,p} \in \{0,1\}$.

Other variables In the model are used also other three continuous variables, with lower bound = 0: d_i(Imbalance variable for each team),*signature_week* and *signature_period*. They are used to evaluate some constraint and not for branching, their use is later explained.

4.2 Objective function

Two operational modes are implemented:

- **Feasibility mode**: no objective (dummy objective 0) — find any feasible schedule given constraints.
- **Optimization/Balanced mode**: minimize the total home/away imbalance across teams. Let

  $$H_i = \text{number of home matches of team } i, \qquad A_i = \text{number of away matches of team } i.$$

  We introduce $d_i \geq 0$ and linearize the absolute imbalance by

  $$-d_i \leq H_i - A_i \leq d_i,$$

  and minimize $\sum_i d_i$. Since in this way the minimal value reachable for the function is equal to n, in the code is used this variant $|H_i - A_i| \leq 1 + d_i$ that allows the solvers to reach 0, so it stops when it finds an optimal solution (obj = 0) and not at the end of the time limit. The objective is:

  $$\min \sum_{i \in T} d_i.$$

4.3 Constraints

4.3.1 Main problem constraints

Let the set of unordered pairs be $E = \{(i,j) : i < j\}$. The main linear constraints are:

(1) One match per (week,period). For every $w \in W$, $p \in P$,

$$\sum_{\substack{i,j \in T \\ i \neq j}} x_{w,p,i,j} = 1 \qquad \text{or} \qquad \sum_{(i,j) \in E} x_{w,p,i,j} = 1$$

if using oriented variables (base), or for the unordered encoding (i¡j).

If using the *preprocessed* version, the constraint will be to assign each precomputed pair to exactly one period:

$$\sum_{p \in P} y_{w,k,p} = 1.$$

This ensures each precomputed pair in each week is placed in exactly one period.

(2) Each pair meets exactly once. For every pair (i, j),

$$\sum_{w \in W} \sum_{p \in P} \big(x_{w,p,i,j} + x_{w,p,j,i}\big) = 1 \qquad \text{or} \qquad \sum_{w \in W} \sum_{p \in P} x_{w,p,i,j} = 1$$

in oriented encodings, or in the unordered encoding.

(3) Each team plays exactly once per week. For every team $t \in T$ and week $w \in W$,

$$\sum_{p \in P} \sum_{j \in T,\, j \neq t} \big(x_{w,p,t,j} + x_{w,p,j,t}\big) = 1$$

(or in $i < j$ notation, sum over the unordered pairs that contain t and all periods equals 1).

(4) Period cap (balance of periods). For every team $t \in T$ and period $p \in P$,

$$\sum_{w \in W} \sum_{j \in T,\, j \neq t} \big(x_{w,p,t,j} + x_{w,p,j,t}\big) \leq 2,$$

or, in $i < j$ encoding, sum over unordered pair variables that include t across weeks ≤ 2.

If using the *preprocessed* version, the constraint will be:

$$\sum_{(w,k) \in \text{appearance\_map}[t]} y_{w,k,p} \leq 2,$$

.

(Permutation constraint) Used only for *preprocessed* version. Eeach period has exactly one pair. For all w, p:

$$\sum_{k=0}^{p-1} y_{w,k,p} = 1.$$

This enforces that, for each week, the y variables form a permutation matrix: each period hosts exactly one of the precomputed pairs.

(Linking orientation) For $i < j$ and for *preprocessed* version when balancing/optimal is enabled, we have:

$$h_{w,p,i,j} \leq x_{w,p,i,j} \quad \forall w, p, i, j, \qquad \text{or} \qquad h_{w,k,p} \leq y_{w,k,p} \quad \forall w, k, p.$$

So when $x = 1$ or $y = 1$, h may be 0 or 1 to choose orientation; if $x = 0$ then h must be 0. If pair k is placed at (w, p) then $h_{w,k,p} = 1$ indicates the *first* team in the precomputed pair is at home; otherwise the second team is at home.

(Imbalance linearization.) For each team i, when balancing/optimal is enabled:

$$-H_i + A_i \leq 1 + d_i, \qquad H_i - A_i \leq 1 + d_i.$$

4.3.2 Symmetry breaking

Symmetry arises from permuting week labels, period labels and flipping home/away orientation globally. We provide optional safe symmetry-breaking constraints controlled by flags `A,B,C,D`:

A (anchor). Anchor team k (default $k = 1$) to occupy a match in $(w = 1, p = 1)$:

$$\sum_{\{i,j\}\ni k} x_{1,1,i,j} \; = \; 1.$$

B (fix week 1 composition). Fix the set of unordered pairs appearing in week 1 to the circle-method composition: for each pair (a, b) fixed,

$$\sum_{p\in P} x_{1,p,a,b} = 1.$$

C (canonical order of weeks). Define a continuous signature variable *signature_week* $= \sigma_w^{\text{week}}$ for each week (weighted sum of team ids participating) and enforce

$$\sigma_w^{\text{week}} \leq \sigma_{w+1}^{\text{week}}, \quad \forall w,$$

to order weeks to one canonical representative.

D (canonical order of periods). Similarly define and *signature_period* $= \sigma_p^{\text{period}}$ and enforce $\sigma_p^{\text{period}} \leq \sigma_{p+1}^{\text{period}}$.

Other symmetry braking, like the permutation of teams of possible solution, are taken into consideration by creating the variables in a fixed way, like for the unordered pairs (`i<j`) and `preprocessed` models.

4.3.3 Warm-Start

For the CBC solver there is the possibility to build MIP warm-starts (MIPStart) by setting initial values for selected binary variables corresponding to the preprocessed schedule. And i tested this implementations:

- week1: set only the first week.
- bal_full (or for preprocessing version random_half): set half of the possible solution found by circol method.
- half_full (or for preprocessing version random_full): set all slots of the solution found.

4.4 Validation

Before starting is important to know that all the tables showed underneath are exstract from the source/MIP/TABLES_summeryze.txt and that the behaviour of CBC observed and documented:

1. **Random seed impact.** CBC does not change by default seed (but uses one with value 1234567): If we set the seed to 0 we achieve a certain degree of randomness since it starts to use time as value for the seed. Using this method, since internal heuristics, probing, cut selection and randomized tie-breaking depend on the random seed, we achieve widely varying runtimes for the same exact model (run MIP $v5$ to have similar result).
2. **Warm-start** A lot of times when we try to use Warm-start CBC may ignore the solutione given, also if it is partial and 100% right. This because it finds that some constraints are not respected, since not all variables have been instantiated. Also this happend because of all the internal mechanisms that CBC does.

Each experimental setting was run with 10 different seeds, one of which was always random.
(0,1234567,26,42,262626,424242,878641,5656565,1756566010,948486489).
If no time interval is reported, the solver reached the time limit of 300 seconds.
The experiments stop at the largest n for which a solution is found. If not explicitly stated, presolve is assumed to be set to `True`.

Table 3: Comparison of v1, v2, and v3 across different n values.

n	v1 (base)		v2 ($i \neq j$)		v3 ($i < j$)	
	T	F	T	F	T	F
Balanced						
12	7/10	4/9 opt + 1/9 non	6/10	3/10	10/10 (10–88s)	9/10
14	0/10	0/10	0/10	0/10	3/10 (97–276s)	4/10
Feasible						
12	10/10 (4–108s)	10/10 (4–108s)	10/10 (3–108s)	10/10 (4–108s)	10/10 (10–60s)	10/10 (1–35s)
14	5/10 (2–279s)	4/10 (4–213s)	5/10 (2–279s)	4/10 (78–213s)	2/10 (21–229s)	3/10 (4–162s)

4.4.1 Test 1

Table 3 illustrates how the choice of formulation and preprocessing affects both feasibility and balanced_optimization versions adn shows the limitations of the three models.
Key observations include: disabling presolve sometimes improves v3-feasible, while v2 does not provide significant benefits (run MIP $v2$ that has the same parameters of v1-v3 to make a comparison).
Version v3 performs well for balanced optimization, whereas v1 is superior for feasibility (*realtive json files are stored in res/MIP/MIP_old/table_1*).

4.4.2 Test 2

The second test (*realtive json files are stored in res/MIP/MIP_old/table_2*) aim to find the best additional constraint to have better performance from the models:
Table 4 shows that the additional constraint D should be removed, since it does not improve results. In fact, v3-balanced performs better without symmetry.
Below are reported also the best combinations founded:

Table 4: Relevant setups.

n	v3-balanced (CBC)
12	10/10
14	2/10 (1/10+non-optimal)

n	v1-feasible (CBC + B)
12	10/10 (1–26s)
14	5/10 (80–197s)

D	v1 (feasible)	v3 (balanced)
12	0/5	0/5

4.4.3 Test 3

Experiments aims to see if adding warm-starts to the feasible version with v1 or v3 let us found better results.
We found, as hypothesized, that fixing Week 1 as constraint is worst then using it as warm start.
For v3-balanced, warm-starts improved stability and solution times for $n = 14$, though disabling presolve degraded performance.
For feasible models, v1 with warm-start Week 1 remained the most effective, particularly for $n = 14$. (*realtive json files are stored in res/MIP/MIP_old/table_3*)

Table 5: Performance of version v3 with objective *balanced* using warm_start = week1. Reported values correspond to average, minimum, and maximum solution times (in seconds) over the specified number of runs.

n	Presolve	Mean (s)	Min (s)	Max (s)	Count
12	True	41.70	20.00	88.00	10
14	True	154.75	85.00	273.00	4
12	False	55.10	17.00	132.00	10
14	False	135.00	53.00	217.00	2

4.4.4 GLPK

CBC showed strong sensitivity to preprocessing and warm-start hints. In contrast, GLPK (which uses a random seed by default) produced more stable results across runs (*realtive json files are stored in res/MIP/MIP_old/test_4*). I tried different flags of GLPK (– cuts, –goromy, –clique – dual –prima (defoult –simplex)) and the best configurations are: feasible → dual + B (17s, $n = 12$), balanced → dual + cuts + A ($n = 10$).
(run MIP $v6$ to see different behaviors of GLPK)

4.4.5 Preprocessing

The preprocessing model is the best of all the previows models both for CBC/GLPK and feasible/optimal versions (*realtive json files are stored in res/MIP/MIP_old/test_5*).

4.4.6 Best models

CBC_{version}_{objective}_{warm_start}_{sym_flags}_{seed}
GLPK_{version}_{objective}_{flagGLPK}_{sym_flags}_{seed}

- `v1_f_C_w1` = `CBC_base_fea_week1__42`;
- `v3_b_C_w1` = `CBC_i<j_bal_week1__878641`;
- **v4_b_C_rh_a** = CBC_prepro_anc_bal_ran_half_424242;
- **v4_f_C_w1_a** = CBC_prepro_anchor_fea_week1_262626;
- `v1_f_G_d_B` = `GLPK_base_fea_dual_B_26`;
- `v3_b_G_dc_A` = `GLPK_i<j_bal_dual_cuts_A_26`;
- `v4_b_G_d_a` = `GLPK_preproc_bal_dual_anchor_26`;
- `v4_f_G_d_a` = `GLPK_preproc_fea_dual_anchor_26`.

Table 6: optimal

ID	v3_b_C_w1	**v4_b_C_rh_a**	v3_b_G_dc_A	v4_b_G_d_a	v1_f_C_w1	**v4_f_C_w1_a**	v1_f_G_d_B	v4_f_G_d_a
4	inf	inf	inf	inf	inf	inf	inf	inf
6	**0**	**0**	**0**	**0**	0s	0s	0s	0s
8	**0**	**0**	**0**	**0**	0s	0s	0s	0s
10	**0**	**0**	**0**	**0**	4s	0s	6s	0s
12	**0**	**0**	–	**0**	3s	2s	18s	2s
14	**0**	**0**	–	–	5s	8s	–	101s
16	–	**0**	–	–	–	1s	–	–

The best performances are obtain with the preprocessed model(v4) with anchor (symmetry braking B) for the optimize/balanced (with warmstart = random_half) and for the feasible version (with warmstart = random_half).

5 Conclusions

In conclusion, aside from the use of symmetry-breaking constraints, two factors proved decisive in improving our results. First, the adoption of the circle method as a preprocessing step provided a strong structural foundation, ensuring feasibility and greatly reducing the search space across all approaches. Second, within the CP framework, the subdivision of the scheduling task from the home/away assignment problem proved crucial, as it isolated complexity into more manageable components and improved solver efficiency.

For the MIP approach, the combination of preprocessing with warm-start strategies further stabilized performance and accelerated convergence, particularly in the balanced optimization setting. On the SMT side, the model with preprocessing combined with the Z3 solver delivered some of the most impressive outcomes, managing to optimize schedules up to n = 18.

References

Barrett, C., P. Fontaine, and C. Tinelli (2016). The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`.

Kendall, G., S. Knust, C. C. Ribeiro, and S. Urrutia (2010). Scheduling in sports: An annotated bibliography. *Computers & Operations Research 37*(1), 1–19.

Lambrechts, E., A. M. Ficker, D. R. Goossens, and F. C. Spieksma (2018). Round-robin tournaments generated by the circle method have maximum carry-over. *Mathematical Programming 172*(1), 277–302.

Larson, J., M. Johansson, and M. Carlsson (2014). An integrated constraint programming approach to scheduling sports leagues with divisional and round-robin tournaments. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 144–158. Springer.