1 SMT Model

1.1 Decision Variables

Let n be the number of teams, and let $W = \{1, \dots, n-1\}$ denote the set of tournament weeks. Each week is divided into $n/2$ periods. Let $\mathcal{M}_w$ be the set of matches scheduled in week $w \in W$.

We define an integer decision variable:

$$p_{w,i,j} \in \{1, \dots, n/2\}$$

for each match between any pair of teams (i, j), where $(i, j) \in \mathcal{M}_w$, indicating the period during which this match takes place in week w.

The model is encoded using the theory of Linear Integer Arithmetic (LIA), extended with pseudo-Boolean constraints for cardinality.

1.1.1 Pre-solving with the Circle Method

We avoid additional decision variables by precomputing which teams (i, j) play in each week w using the fast and classical *circle method* for round-robin tournaments [1].

One team (the pivot) is fixed, and the other $n - 1$ teams are placed in a circle. Each week, the pivot plays a rotating team, while the remaining teams are paired by connecting opposite positions in the circle. This rotation continues for $n - 1$ weeks. The construction guarantees:

- Each team plays once per week;
- No match is repeated;
- The total number of matches is $\frac{n(n-1)}{2}$.

Example for $n = 6$:

$$\text{Week 1: } (6, 1), (2, 5), (3, 4) \quad \text{Week 2: } (6, 2), (3, 1), (4, 5), \dots$$

In practice, this greatly reduces the number of SMT variables and constraints. Thanks to pre-solving, we can solve instances with up to $n = 18$ teams—and usually even $n = 20$—within the 5-minute limit. This is 2 to 4 teams more than our first SMT models that generated all possible matches as decision variables.

1.2 Objective Function

For the sake of solving the home-away balancing task, there is a simple mathematical algorithm. It does not need optimization with SMT. If (i,j) is a game between any two teams i and j then the modulo arithmetic's algorithm is this:

$$i, j \in \{1, \dots, n\}, \quad d \equiv j - i \pmod{n},\ 1 \le d \le n-1, \qquad \text{home}(i, j) = \begin{cases} i, & d < \frac{n}{2}, \\ j, & d \ge \frac{n}{2}, \end{cases}$$

By construction, it achieves an absolute disbalance of home/away games equal to 1 per each team. In total, the sum of the absolute disbalance values will be n. Theoretically, it is minimal for even n.

We use this mathematical approach in the decisional version, so the json results for the decisional version are also home/away optimal.

But to study the solver overhead of enforcing this balance via optimization, we performed a dedicated experiment. We define for team t - absolute difference between home and away games as:

$$\mathit{abs\_diff}_t \;=\; \big|(n-1) \;-\; 2\cdot \mathit{away\_count}_t\big|$$

We introduced the global imbalance metric

$$\texttt{sumDif} \;=\; \sum_{t=1}^{n} \mathit{abs\_diff}_t$$

as our objective function. Also, we add the constraint

$$\texttt{sumDif} \;\geq\; n$$

We add following additional symmetry-breaking constraint (w.r.t to team numbering) on the per-team imbalances:

$$\mathit{abs\_diff}_{t_1} \;\geq\; \mathit{abs\_diff}_{t_2} \;\geq\; \cdots \geq\; \mathit{abs\_diff}_{t_n},$$

and then instructed Z3 to minimize

$$\max_t \mathit{abs\_diff}_t = abs\_diff_0$$

In our experiments, it appeared that instead of minimizing `sumDif`, it is better to minimize $\max_t \mathit{abs\_diff}_t$ objective, as the optimization becomes almost an order of magnitude faster.

1.3 Constraints

Domain Constraints. Each decision variable must be assigned a valid period:

$$\forall (w,i,j): \quad 1 \leq p_{w,i,j} \leq n/2$$

Unique Match per Period per Week. For each week $w \in W$ and period $k \in \{1,\ldots,n/2\}$, exactly one match must be assigned to that period:

$$\sum_{(i,j)\in\mathcal{M}_w} [p_{w,i,j} = k] = 1$$

Period Load per Team. Each team must appear at most twice in each period across the tournament:

$$\forall t \in [n], \forall k \in \{1,\ldots,n/2\}: \qquad \sum_{(w,i,j):t\in\{i,j\}} [p_{w,i,j} = k] \leq 2$$

Symmetry Breaking Constraints

Fixed Periods in First Week. To reduce the search space, we fix the period assignment of all matches in the first week:

$$\forall k \in \{1, \ldots, n/2\}, \quad p_{1,i_k,j_k} = k$$

where (i_k, j_k) is the k-th match in a lexicographically sorted list of matches in week 1. This removes the permutation symmetry for the period labels.

In our tests more complicated constraints - like lexicographical constraints on period assignment vectors for chosen pairs of teams - did not bring any noticeable additional improvements, as well as adding redundant constraints.

1.4 Validation

Solver. The model was implemented in Python using the Z3 SMT solver. A custom tactic pipeline was used:

```
Then('card2bv', 'smt')
```

to enable pseudo-Boolean cardinality constraints translation into bit-vector constraints for speed.

Experimental Setup.

- **Language:** Python 3.11
- **Solver:** Z3 v4.15.2
- **Hardware:** Intel Xeon CPU of Google Colab
- **Timeout:** 300 seconds

Experimental Results (Decision version, SMT in Z3)

We tested the SMT model (without home-away optimization) using Z3 on increasing values of n, with and without symmetry breaking. All runs were limited to 300 seconds. The results in Table 1 show that enabling symmetry breaking significantly improves performance, especially for larger instances.

We observed that while symmetry breaking introduced a slight overhead for small n, it consistently reduced runtime as n increased, especially beyond $n = 16$. Without SB, solving $n = 20$ takes more than three minutes, while the version enabled with SB is completed in less than 45 seconds.

Though we need to mention that for a fixed n there is still some variation between runs - due to the solver's heuristic choices adding some randomness. In rare cases $n = 20$ might take more than 5 minutes.

Table 1: Runtime in seconds for SMT (decision version), Z3 solver, with and without symmetry breaking (SB).

Number of Teams	**SB Enabled**	**SB Disabled**
10	0.04	0.09
14	0.92	0.43
16	8.21	1.50
18	9.14	15.65
20	44.74	181.71

Experimental Results (Optimization version, SMT in Z3)

We extended the SMT model to solve the optimization version of the problem, where the objective is to minimize the maximal absolute discrepancy in home/away balance. For n teams, the theoretical optimum for the sum of absolute discrepancies is n (each absolute discrepancy equal to 1), and this value was reached in all tested instances.

Table 2 reports the total solving time (in seconds) for various team sizes, both with and without symmetry breaking (SB). All runs were performed with Z3 under a 5-minute timeout. In all cases, the solver found and proved optimal solutions within the time limit.

Table 2: Runtime in seconds for SMT (optimization version), Z3 solver, with and without symmetry breaking (SB). The objective function, $\max_t abs_diff_t$, in every case was optimal, and the value is shown in bold.

Number of Teams	**SB Enabled**	**SB Disabled**
10	0.23|**1**	0.41|**1**
12	1.25|**1**	1.13|**1**
14	6.57|**1**	5.35|**1**
16	50.75|**1**	53.23|**1**
18	73.31|**1**	171.56|**1**
20	130.75|**1**	260.65|**1**

We observe that for smaller instances, symmetry breaking has a small impact or slight overhead. However, for larger values of n, symmetry breaking leads to significantly faster convergence to the optimal solution, nearly halving the runtime in the case of $n = 20$.

Though there is a noticeable variance due to solver heuristic choices, the observations made above are true for the majority of the re-runs, if made.

References

[1] Dominique de Werra. Scheduling Round-Robin Tournaments: An Overview. *Discrete Applied Mathematics*, 91(1–3):241–277, 1999.