1 SAT Model

1.1 Decision Variables

Let n be the (even) number of teams. We define the sets of teams, weeks, and periods per week as

$$T = \{0, \ldots, n-1\}, \quad W = \{0, \ldots, n-2\}, \quad P = \left\{0, \ldots, \frac{n}{2} - 1\right\}.$$

Let $\mathcal{M}_w$ be the set of matches scheduled in week $w \in W$.

To encode the Sports Tournament Scheduling (STS) problem, we define the following decision variables:

1. **match_period_vars**$[i, j, p] \in \{\text{True}, \text{False}\}$, for all $i, j \in T$ with $i < j$, and $p \in P$.
   This variable is true if and only if team i plays against team j during period p. The week for each match-up (i, j) is precomputed (see Section 1.1.1) and fixed.

2. **home_vars**$[i, j] \in \{\text{True}, \text{False}\}$, for all $i, j \in T$ with $i < j$.
   This variable is true if and only if team i plays at home against team j.

1.1.1 Pre-solving with the Circle Method

To avoid introducing additional decision variables for the weeks, we precompute the match-ups for each week w using the classical *circle method* for round-robin tournaments [1]. We use a dictionary **pair_to_week** to store the fixed week for each pair (i, j).

This method fixes one team (the pivot) and arranges the other $n-1$ teams 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, guaranteeing a valid round-robin schedule.

1.2 Objective Function

The goal of the STS problem is to minimize the maximum home-away imbalance among all teams:

$$\min \max_{t \in T} |H_t - A_t|,$$

where H_t is the number of home games for team t, and A_t is the number of away games for team t.

Since SAT solvers handle only boolean variables, we solve this optimization problem via a *binary search* on the maximum allowed imbalance k.

For each team $t \in T$, the number of home games H_t is defined as

$$H_t = \sum_{\substack{(i,j) \in \text{pair\_to\_week} \\ p \in P}} \begin{cases} 1 & \text{if } t = i \wedge \text{match\_period\_vars}_{i,j,p} \wedge \text{home\_vars}_{i,j} \\ 1 & \text{if } t = j \wedge \text{match\_period\_vars}_{i,j,p} \wedge \neg\text{home\_vars}_{i,j} \\ 0 & \text{otherwise} \end{cases}$$

To enforce that the home-away imbalance for each team t does not exceed k, we use two pseudo-boolean constraints. Let NUM_GAMES $= n - 1$. For a given k, the constraints are:

1. $H_t \geq$ lower_bound, where lower_bound $= \left\lceil \frac{\text{NUM\_GAMES}-k}{2} \right\rceil$,
2. $H_t \leq$ upper_bound, where upper_bound $= \left\lfloor \frac{\text{NUM\_GAMES}+k}{2} \right\rfloor$.

These bounds ensure that the number of home games for each team respects the maximum imbalance k. Since the maximum number of matches a team can play is $|W|$, which is odd, then the optimal value we want to achieve is exactly $k = 1$.

1.3 Constraints

Thanks to the *circle method*, some constraints are inherently satisfied:

1. Each team plays every other team exactly once.
2. Each team plays exactly once every week.

This reduces the constraints needed, leaving the solver to decide only the specific period for each match (i, j). In particular, we implemented the following constraints:

Each match is assigned to exactly one period:

$$\forall (i,j) \in \text{pair\_to\_week} : \sum_{p \in P} \text{match\_period\_vars}_{i,j,p} = 1$$

Each period in each week contains exactly one match:

$$\forall w \in W, \forall p \in P : \sum_{(i,j) \in \mathcal{M}_w} \text{match\_period\_vars}_{i,j,p} = 1$$

Each team plays at most twice in the same period:

$$\forall t \in T, \forall p \in P : \sum_{\substack{(i,j) \in \text{pair\_to\_week} \\ t \in \{i,j\}}} \text{match\_period\_vars}_{i,j,p} \leq 2$$

Symmetry Breaking Constraints

To reduce the search space and speed up the solver, we implemented the following symmetry breaking constraints:

SB1: Match between teams 0 and $n-1$ is in the first period:

$$\text{match\_period\_vars}_{0,n-1,0} = 1$$

This fixes the match between the pivot team 0 and team $n - 1$ in period 0, breaking rotational symmetry.

SB2: Team 0's home/away pattern is fixed:

$$\forall (i,j) \in \text{pair\_to\_week with } 0 \in \{i,j\} : \begin{cases} \text{home\_vars}_{i,j} & \text{if } w(i,j) \text{ is even and } i = 0 \\ \neg\text{home\_vars}_{i,j} & \text{if } w(i,j) \text{ is even and } j = 0 \\ \neg\text{home\_vars}_{i,j} & \text{if } w(i,j) \text{ is odd and } i = 0 \\ \text{home\_vars}_{i,j} & \text{if } w(i,j) \text{ is odd and } j = 0 \end{cases}$$

This fixes team 0's home/away assignment pattern, eliminating symmetries from flipping all home/away statuses.

SB3: Lexicographical ordering of matches in week 0:

$$\forall a \in \{0, \dots, |\mathcal{M}_0| - 2\} : \quad \text{lex\_less}(V_a, V_{a+1})$$

This enforces a lexicographical order on the period assignments for matches in week 0, preventing symmetric solutions.

Encoding Methods

We implemented all the encoding methods covered in class:

1. For the *exactly-one* constraint: Pairwise, Bitwise, Sequential, and Heule encodings.
2. For the *at-most-k* constraint: Pairwise, Sequential, and Totalizer encodings.

As mentioned, we employed the **Totalizer encoding** [2] to efficiently model cardinality constraints. This encoding builds a balanced binary tree of adders and is known for scalability and efficiency in SAT formulations. It introduces $O(n \log n)$ auxiliary variables and up to $O(n^2)$ clauses in the worst case.

1.4 Validation

The model was implemented in Python using Z3's API. All experiments were conducted respecting the given timeout of 300 s and with an increasing number of instances (i.e., number of teams, n).

In the following, we present tables and plots to compare the Z3 model using different encoding techniques, both with and without symmetry breaking constraints.

1.4.1 Experimental Results (Decision version)

The decision version of the STS problem was tested using the HEULE encoding for the *exactly-one* constraint and the SEQUENTIAL encoding for the *at-most-k* constraint.

The results in Table 1 show that the symmetry breaking constraints did not significantly speed up the solver in finding a solution.

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

Number of Teams	**heule-seq + SB**	**heule-seq w/out SB**
12	0.0	0.0
14	1.0	0.0
16	10.0	14.0
18	19.0	17.0
20	217.0	248.0

1.4.2 Experimental Results (Optimization version)

The optimization version of the Sports Tournament Scheduling (STS) problem was solved using a binary search approach on the objective function value. The problem was encoded into SAT instances, using a combination of different encodings for the *exactly-one* and *at-most-k* constraints:

1. Pairwise + Pairwise encodings
2. Heule + Sequential encodings
3. Heule + Totalizer encodings

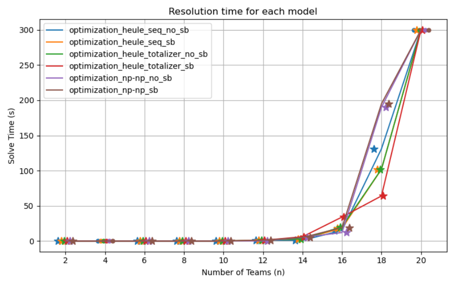
As shown in Table 2, the results confirm the expected performance of the encodings. The Heule + Totalizer combination generally proved to be the most efficient, particularly for larger instances where the Totalizer encoding's logarithmic complexity shines. Conversely, the Pairwise + Pairwise approach, known for its larger number of clauses, was the slowest.

Regarding the use of symmetry-breaking constraints (SB), their impact on performance varied. For small instances ($N \leq 16$), the overhead of the constraints sometimes led to a slight increase in runtime. However, for larger instances ($N > 16$), the symmetry breaking proved effective at guiding the solver to an optimal solution faster, significantly reducing the search space. This is evident by comparing the runtimes for $N = 18$ and $N = 20$.

The table also shows that all tested configurations successfully found an optimal solution for $N \leq 18$. For $N = 20$, only the more efficient encodings (Heule + Sequential and Heule + Totalizer) with symmetry breaking enabled were able to find a feasible, though suboptimal (value 10), solution within the time limit. This highlights the importance of choosing a robust encoding and constraint set for tackling complex problem instances.

Table 2: SAT results. Results are of the type ”time|**optimal value**” or ”time|suboptimal value”.

N	np-np + SB	np-np w/out SB	heule-seq + SB	heule-seq w/out SB	heule-tot + SB	heule-tot w/out SB
10	0|**1**	0|**1**	0|**1**	0|**1**	0|**1**	0|**1**
12	1|**1**	0|**1**	1|**1**	1|**1**	1|**1**	1|**1**
14	5|**1**	5|**1**	3|**1**	1|**1**	6|**1**	3|**1**
16	19|**1**	12|**1**	18|**1**	15|**1**	34|**1**	20|**1**
18	195|**1**	190|**1**	102|**1**	131|**1**	64|**1**	102|**1**
20	N/A	N/A	300|10	N/A	300|10	N/A



(a) Resolution time

Figure 1: Runtimes for the optimization version of the Sports Tournament Scheduling (STS) problem. Data points are marked to indicate the solution status: a star (⋆) indicates that a solution (optimal or sub-optimal) was found, while a circle (●) indicates that no solution was found within the time limit.

References

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

[2] Olivier Bailleux and Yacine Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In: *Principles and Practice of Constraint Programming (CP 2003)*, Lecture Notes in Computer Science, 2003.