Sports Tournament Scheduling: A Multi-Paradigm Optimization Approach

Leonardo Tassinari, leonardo.tassinari6@studio.unibo.it
Francesco Zattoni, francesco.zattoni2@studio.unibo.it

July 2025

1 Introduction

This report presents a comprehensive study of the Sports Tournament Scheduling (STS) problem, focusing on the development and comparison of multiple optimization models. The aim is to provide a unified framework for modeling, analyzing, and solving the STS problem, enabling a fair and systematic evaluation of different solution paradigms. All models considered in this work share a common formalization, which is described in this section.
The repository is available at `https://github.com/leo-tasso/STS`.

1.1 Common Model Formalization

Input Parameters. The STS problem is defined by the following parameters, which are common to all models:

- n: Number of teams (even integer)
- $w = n - 1$: Number of weeks
- $p = n/2$: Number of periods (matches per week)
- Teams are indexed by $t \in \{1, 2, \ldots, n\}$
- Weeks are indexed by $w \in \{1, 2, \ldots, n-1\}$
- Periods are indexed by $p \in \{1, 2, \ldots, n/2\}$

Objective Variable and Bounds. For the optimization variant, the objective is to minimize the maximum imbalance in home and away games for any team:

$$M = \max_{t \in \{1,\ldots,n\}} |h_t - a_t|$$

where h_t and a_t denote the number of home and away games played by team t, respectively. The upper bound for M is $n-1$, even if the practical upper bound is lower, because it's impossible that every team plays only away games or home games. The theoretical lower bound for M is 1 for even n, since h_t and a_t cannot be equal.

Constraints. All models enforce the following core constraints:

1. Each pair of teams plays exactly once during the tournament.
2. Each team plays exactly once per week.
3. Each period in each week hosts exactly one match.
4. No team plays against itself.
5. Each team appears in the same period at most twice across all weeks.

Pre-processing and Symmetry Breaking. To improve solver efficiency, all models may include pre-processing steps such as:

- Translating the model into a solver-independent language (such as DIMACS or SMT-LIB), which may require additional pre-processing time before the actual solving phase.
- Fixing the schedule of the first week to break team symmetries.
- Lexicographic ordering of weeks and/or periods to break week and period symmetries.
- Adding implied constraints (e.g., total matches per team).

Validation Each test is executed five times, using a timeout of 300 seconds, on the same hardware: a Ryzen 5 5600X CPU with 32 GB of RAM, running inside a Docker container based on Ubuntu 22.04. Different solvers and approaches are evaluated, including variations in active constraints. The result tables report both the time in seconds taken for the decision version and the objective value found where the optimization version is used (CP, Z3 both SAT and SMT, MIP). Each cell contains `decision_seconds`|`opt_obj_value`, if an optimization version is defined, otherwise only the `decision_seconds`.

2 CP model

2.1 Decision Variables

The MiniZinc CP model represents the tournament schedule using two primary arrays of integer decision variables:

- `home[`w`, `p`]`: For each week w and period p, `home[`w`, `p`]` is an integer variable indicating the team assigned as the home team in that slot.
- `away[`w`, `p`]`: For each week w and period p, `away[`w`, `p`]` is an integer variable indicating the team assigned as the away team in that slot.

Both variables take values in $\{1, \ldots, n\}$ and together define the assignment of teams to each match in the tournament schedule. All additional variables and constraints in the model are defined in terms of these primary decision variables, as described in the common formalization.
Another design of the decision variables was tested: `matches`, an array for each possible match ID that takes values in $\{1..slots\}$, $slots = (n-1) \times (n/2)$, but it led to a more complex implementation with not-so-evident improvements only for $n = 12$. Also, the addition of channeling constraints between `matches`, `home` and `away` didn't help.

2.2 Objective Function

The objective variable and its theoretical bounds are described in Section 1. In the MiniZinc model, the objective is implemented as the variable `max_diff`, which represents the maximum absolute difference between the number of home and away games for any team:

$$\texttt{max\_diff} = \max_{t \in \text{Teams}} |\, \texttt{home\_count}[t] - \texttt{away\_count}[t]\, |$$

where the auxiliary variables `home_count`[t] and `away_count`[t] are defined as follows:

- `home_count`[t]: the total number of times team t appears as the home team across all weeks and periods,
$$\texttt{home\_count}[t] = \sum_{w \in \text{Weeks}} \sum_{p \in \text{Periods}} [\texttt{home}[w,p] = t]$$
- `away_count`[t]: the total number of times team t appears as the away team across all weeks and periods,
$$\texttt{away\_count}[t] = \sum_{w \in \text{Weeks}} \sum_{p \in \text{Periods}} [\texttt{away}[w,p] = t]$$

The auxiliary variables `home_count`[t] and `away_count`[t] are computed efficiently using the `global_cardinality` constraint in MiniZinc.

The model minimizes `max_diff` using the following directive:

```
solve minimize max_diff;
```

2.3 Constraints

Main Problem Constraints

- **Each pair of teams plays exactly once.** For every unordered pair of distinct teams (i, j), there must be exactly one match where i plays against j (either as home or away):
$$\sum_{w \in \text{Weeks}} \sum_{p \in \text{Periods}} ([\texttt{home}[w,p] = i \land \texttt{away}[w,p] = j] + [\texttt{home}[w,p] = j \land \texttt{away}[w,p] = i]) = 1$$
- **Each team plays exactly once per week.** In every week w, each team must appear exactly once, either as home or away. This is enforced by requiring that all teams assigned in week w are distinct:

  The set $\{\texttt{home}[w,p],\ \texttt{away}[w,p] : p \in \text{Periods}\}$ contains all teams without repetition.

  This constraint is implemented using `all_different` for every week w.
- **Each team appears in the same period at most twice.** To model this constraint, we define the variable `period_count`[t, p] as follows:
$$\texttt{period\_count}[t,p] = \sum_{w \in \text{Weeks}} ([\texttt{home}[w,p] = t] + [\texttt{away}[w,p] = t])$$

  This expression is implemented in MiniZinc using the `global_cardinality` constraint, used to enforce the final constraint as:
$$\texttt{period\_count}[t,p] \leq 2$$

Implied Constraints These constraints are semantically redundant, but they can be useful to reduce the search space.

- **Total matches per team:** Each team t must play exactly $n-1$ matches in total (as home or away):
$$\texttt{home\_count}[t] + \texttt{away\_count}[t] = n-1$$
- **Total period appearances per team:** Each team t must appear in some period exactly $n-1$ times over the tournament, exploiting the previously defined variable `period_count`:
$$\sum_{p \in \text{Periods}} \texttt{period\_count}[t,p] = n-1$$

Symmetry Breaking Constraints The STS problem exhibits several symmetries (e.g., relabeling teams, reordering weeks or periods) that can lead to redundant search. Symmetry breaking constraints reduce the number of equivalent solutions and improve solver efficiency:

- **Week symmetry:** Weeks can be permuted without changing the problem. To break this symmetry, we impose lexicographic ordering between consecutive weeks:
$$(\texttt{home}[w,1],\dots,\texttt{home}[w,p],\texttt{away}[w,1],\dots,\texttt{away}[w,p]) <_{\text{lex}} (\texttt{home}[w+1,1],\dots,\texttt{away}[w+1,p])$$
for all $w \in \{1,2,\dots,n-2\}$.
- **Period symmetry:** Periods within a week can be permuted. We enforce lexicographic ordering between periods:
$$(\texttt{home}[1,p],\dots,\texttt{home}[w,p],\texttt{away}[1,p],\dots,\texttt{away}[w,p]) <_{\text{lex}} (\texttt{home}[1,p+1],\dots,\texttt{away}[w,p+1])$$
for all $p \in \{1,2,\dots,n/2-1\}$.
- **Team symmetry:** Teams can be relabeled. We fix the first week schedule to a canonical order:
$$\texttt{home}[1,i] = 2i-1, \quad \texttt{away}[1,i] = 2i$$
for all p in periods.
This approach was already used for a similar problem, as shown in [1].

2.4 Validation

To validate the MiniZinc model, we conducted a systematic experimental study using two different solvers: **Gecode** and **Chuffed**. Both solvers were run on the same set of problem instances to allow for a fair comparison. The experiments were designed to assess the impact of symmetry breaking and implied constraints, as well as the effect of different search strategies.

Experimental Design The validation of the MiniZinc model was performed using the two already mentioned solvers with common search strategies: `first_fail` heuristic to select the variable with the smallest domain, `relax_and_reconstruct(85)` to escape local minima by fixing 85% of the found solution and reconstructing the remaining part of it.
One search strategy differs per solver: Gecode employed also the `indomain_random` to assign a randomly chosen value from the selected variable; Chuffed was tested using `indomain_min`, as

`indomain_random` is not supported.
Restart strategies were tested only with Chuffed, even though it did not use `indomain_random`, because applying any restart strategy with Gecode led to worse performance. Among the strategies tested, `restart_luby` showed the best performance in previous experiments and was therefore selected for the final tests.
Experiments were conducted on MiniZinc 2.9.3 with Gecode 6.3.0 and Chuffed 0.13.2.
For each instance, we evaluated the following configurations:

- Chuffed and Gecode with/without symmetry breaking constraints,
- Chuffed and Gecode with/without implied constraints,
- Chuffed and Gecode with/without search strategies.

Also, other strategies were used, but they didn't improve either the performance or the complexity of the implemented model: we tried `bounds` and `domain` to make the solver use specific kinds of constraint propagation; a change in the order of the variables, i.e. swapping the weeks with the periods in both `home` and `away` arrays.

Experimental Results The results obtained using the described search strategies are reported in the following tables and in the plot to assess the effectiveness of symmetry breaking constraints, implied constraints, and search strategies. In the table 1, C. stands for Chuffed and G. stands for Gecode.

ID	C. (all)	C. w/o SB	C. w/o IC	C. w/o SS	G. (all)	G. w/o SB	G. w/o IC	G. w/o SS
6	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**
8	0 | **1**	0 | **1**	0 | **1**	0 | **1**	1 | **1**	9 | **1**	73 | **1**	2 | **1**
10	3 | **1**	4 | **1**	3 | **1**	3 | **1**	8 | **1**	63 | **1**	146 | **1**	`N/A` | `N/A`
12	110 | **3**	16 | **1**	56 | **5**	109 | **5**	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`
14	`N/A` | `N/A`	37 | **13**	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`
16	`N/A` | `N/A`	`N/A` | **15**	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`	`N/A` | `N/A`

Table 1: Results using Chuffed (C.) and Gecode (G.) with the possible combinations of active constraints, both in decision and optimization version.

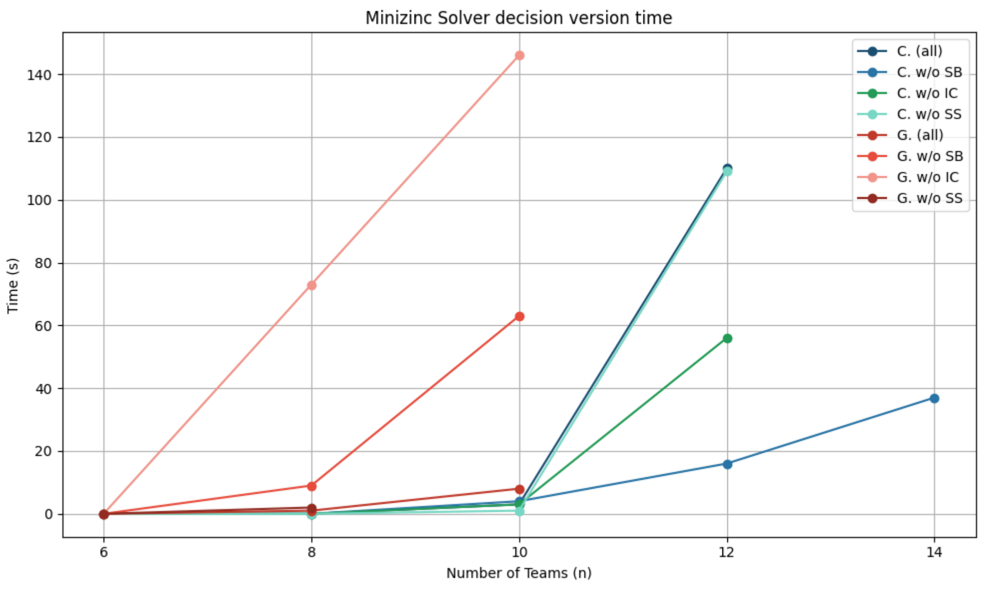

Figure 1: Minizinc solvers: solving time for different constraint configurations.

3 SAT Model

3.1 Decision variables

The core decision variables in the SAT model are 3D arrays:

- `home[`w`, `p`, `t`]`: Boolean variables where `home[`w`, `p`, `t`]` is true if and only if team t plays at home in week w during period p.

  `away[`w`, `p`, `t`]`: Boolean variables where `away[`w`, `p`, `t`]` is true if and only if team t plays away in week w during period p.

Other auxiliary variables are added for the encodings of the cardinality constraints, that will be discussed in Section 3.3.

3.2 Objective Function

To optimize an objective function in SAT we use a *binary search* approach over possible values of the objective. For each candidate value, we encode the constraint "no team has a home/away imbalance greater than d" as SAT clauses, and check satisfiability.
The objective is:

$$\min \max_t |H_t - A_t|$$

Since $H_t + A_t = w$ for all t, minimizing $|H_t - A_t|$ is equivalent to bounding H_t within $[\lfloor w/2 \rfloor - d/2, \lceil w/2 \rceil + d/2]$ for some d.
The sum $H_t = \sum_{w,p}$ `home[`w`, `p`, `t`]` is the number of home games for team t. To enforce the objective, for a given candidate d, we add for each team t:

$$\texttt{min\_home} \leq H_t \leq \texttt{max\_home}$$

where

$$\texttt{min\_home} = \max\left(0, \frac{w-d}{2}\right), \quad \texttt{max\_home} = \min\left(w, \frac{w+d}{2}\right)$$

During the binary search, d becomes smaller, approaching 1, then `min_home` and `max_home` approach $w/2$, leading to a more balanced scheduling.

3.3 Constraints

Main Problem Constraints

- **Each slot has exactly one home and one away team, and they are different:**

$$\forall w, p: \quad \texttt{exactly\_one}(\{\forall t : \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]}\}) \tag{1}$$
$$\forall w, p: \quad \texttt{exactly\_one}(\{\forall t : \texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]}\}) \tag{2}$$
$$\forall w, p, t: \quad \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]} \implies \neg\texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]} \tag{3}$$

  This ensures each slot is filled by one home and one away team, and a team cannot be both in the same slot.

- **Each pair of teams plays exactly once:**

$$\forall i < j: \quad \texttt{exactly\_one}(\{\forall w, p : \texttt{home[}w\texttt{, }p\texttt{, }i\texttt{]} \land \texttt{away[}w\texttt{, }p\texttt{, }j\texttt{]}, \texttt{home[}w\texttt{, }p\texttt{, }j\texttt{]} \land \texttt{away[}w\texttt{, }p\texttt{, }i\texttt{]}\}) \tag{4}$$

  This ensures every pair of teams meets exactly once, with one as home and one as away.

- **Each team plays once per week:**

$$\forall w, t: \quad \texttt{exactly\_one}(\{\forall p : \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]}, \texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]}\}) \tag{5}$$

  Each team appears in exactly one match per week.

- **Period limit: Each team appears in the same period at most twice:**

$$\forall t, p: \quad \texttt{at\_most\_k}(\{\forall w : \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]}, \texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]}\}, 2) \tag{6}$$

  No team is scheduled in the same period more than twice.

Implied Constraints

- **Number of games per team:**

$$\forall t: \quad \texttt{exactly\_k}(\{\forall w, p : \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]}, \texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]}\}, n-1) \tag{7}$$

  Each team plays $n-1$ games (one per week).

- **Total period appearances:**

$$\forall t: \quad \texttt{exactly\_k}(\{\forall w, p : \texttt{home[}w\texttt{, }p\texttt{, }t\texttt{]}, \texttt{away[}w\texttt{, }p\texttt{, }t\texttt{]}\}, n-1) \tag{8}$$

  Each team appears in periods a total of $n-1$ times.

Symmetry Breaking Constraints The lexicographical order constraints used in the CP model are enforced only on the first pair of weeks and periods, because applying them on the whole schedule would lead to an explosion of constraints and to an overall time performance decrease.

- **Weeks:** Lexicographically order the first two weeks to break symmetry:

$$\texttt{lex\_less}([\forall t, p : \texttt{home[0, }p\texttt{, }t\texttt{]}, \texttt{away[0, }p\texttt{, }t\texttt{]}], \tag{9}$$

$$[\forall t, p : \texttt{home[1, }p\texttt{, }t\texttt{]}, \texttt{away[1, }p\texttt{, }t\texttt{]}]) \tag{10}$$

- **Periods:** Lexicographically order the first two periods to break symmetry:

$$\texttt{lex\_less}([\forall w, t : \texttt{home[}w\texttt{, 0, }t\texttt{]}, \texttt{away[}w\texttt{, 0, }t\texttt{]}], \tag{11}$$

$$[\forall w, t : \texttt{home[}w\texttt{, 1, }t\texttt{]}, \texttt{away[}w\texttt{, 1, }t\texttt{]}]) \tag{12}$$

- **Teams:** Fix the assignment in the first week:

$$\forall i : \quad \texttt{home[0, }i\texttt{, }2i\texttt{]} \wedge \texttt{away[0, }i\texttt{, }2i+1\texttt{]} \tag{13}$$

  This removes equivalent solutions due to team relabeling.

Encoding Methods

- **Exactly-one, at-most-k, and exactly-k constraints:** These are encoded using various SAT encodings, selectable in the CLI by the user:
  - `np`: Naive pairwise encoding.
  - `seq`: Sequential counter encoding (used for at-most-k and exactly-k).
  - `bw`: Bitwise encoding for exactly-one.
  - `he`: Heule encoding for exactly-one.

  The sequential counter encoding is used for cardinality constraints (at-most-k, at-least-k, exactly-k) for efficiency. Indeed, in Section 3.4 the validation results report only the ones given the best encoding methods, i.e. the bitwise one for the cardinality constraints with $k = 1$ and the sequential one for constraints with $k > 1$.

3.4 Validation

Model Validation with Z3. The primary implementation of the STS model uses the Z3 SMT solver. All constraints are encoded as Z3 Boolean formulas, and the model is validated by checking satisfiability and extracting a solution using Z3's API.

Solver-Independent Validation via DIMACS. The Z3 model can also be exported to the standard DIMACS CNF format, which is supported by most SAT solvers. This is achieved by:

- Using Z3's `tseitin-cnf` tactic [2] to convert the Z3 Boolean model into CNF.
- Exporting the CNF to DIMACS format, including variable mappings, i.e. the mapping from the assigned integer value to the Z3 boolean variable.

- Parsing the DIMACS file and solving it with external SAT solvers (e.g., Minisat, Glucose) via the PySAT library.
- Decoding the resulting model back into the tournament schedule using the variable mappings.

This approach allows the same model to be validated with different SAT solvers, ensuring correctness and robustness of the encoding.

Experimental Results The following tables and plot show the time performance of the different solvers given the best encoding method:

ID	Z3 (all)	Z3 w/o SB	Z3 w/o IC
6	0 | **1**	0 | **1**	0 | **1**
8	2 | **1**	2 | **1**	0 | **1**
10	8 | **1**	10 | **5**	1 | **1**
12	`N/A` | **6**	`N/A` | **6**	`N/A` | **6**

Table 2: Results using Z3 with the possible combinations of active constraints.

ID	G. (all)	G. w/o SB	G. w/o IC	M.S. (all)	M.S. w/o SB	M.S. w/o IC
6	0	0	0	0	0	0
8	0	1	0	0	1	0
10	4	72	2	4	48	5
12	116	`N/A`	132	49	`N/A`	281

Table 3: Results using Glucose (G.) and MiniSAT (M.S.) with the possible combinations of active constraints.

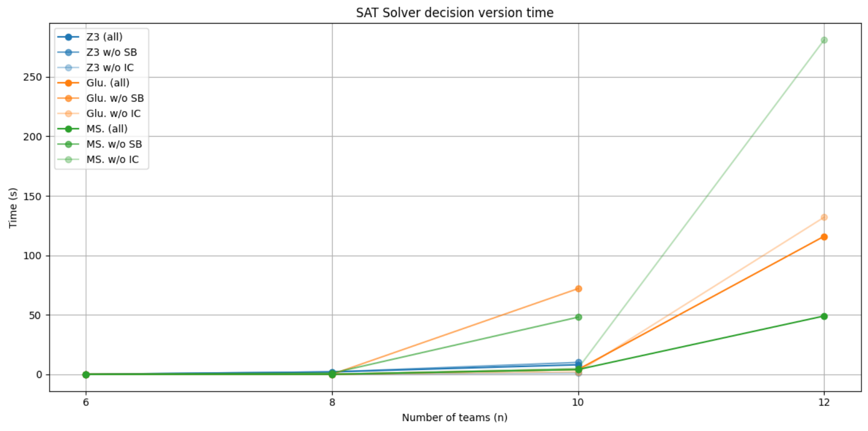

Figure 2: SAT solvers: solving time for different constraint configurations.

4 SMT Model

4.1 Decision Variables

The SMT model employs integer decision variables that naturally express the tournament structure. All variables are defined over the integer sort using the theory of linear integer arithmetic (LIA):

$$\forall w, p : \texttt{home}[w,p] : \textbf{Int}, \quad 1 \le \texttt{home}[w,p] \le n \tag{14}$$
$$\forall w, p : \texttt{away}[w,p] : \textbf{Int}, \quad 1 \le \texttt{away}[w,p] \le n \tag{15}$$

where $\texttt{home}[w,p]$ represents the team playing at home in week w, period p, and $\texttt{away}[w,p]$ represents the team playing away in the same slot.

Auxiliary variables:

$$\forall t : \texttt{home\_count}[t] : \textbf{Int} \tag{16}$$
$$\forall t : \texttt{away\_count}[t] : \textbf{Int} \tag{17}$$
$$\texttt{max\_diff} : \textbf{Int} \tag{18}$$

4.2 Objective Function

The objective function minimizes the maximum home/away imbalance across all teams. In SMT, this is expressed using optimization assertions:

$$\textbf{minimize } \texttt{max\_diff} \tag{19}$$

where `max_diff` is constrained as:

$$\forall t : \quad \texttt{max\_diff} \ge |\texttt{home\_count}[t] - \texttt{away\_count}[t]| \Rightarrow \tag{20}$$
$$\texttt{max\_diff} = \max_t |\texttt{home\_count}[t] - \texttt{away\_count}[t]| \tag{21}$$

The above defined objective function is inserted only in the Z3 solver, since SMT-LIB does not support natively the optimization.

4.3 Constraints

All constraints are expressed using the theory of linear integer arithmetic (LIA).

Main Problem Constraints

- **No team plays against itself:**

$$\forall w, p : \quad \texttt{home}[w,p] \ne \texttt{away}[w,p] \tag{22}$$

- **Each team plays exactly once per week:**

$$\forall w : \quad \textbf{distinct}(\texttt{home}[w,1], \ldots, \texttt{home}[w,n/2], \texttt{away}[w,1], \ldots, \texttt{away}[w,n/2]) \tag{23}$$

- **Each pair of teams plays exactly once:**

$$\forall i, j \text{ with } i < j: \quad \sum_{w=1}^{n-1} \sum_{p=1}^{n/2} \Big(\mathbf{ite}(\texttt{home}[w,p] = i \wedge \texttt{away}[w,p] = j, 1, 0) + \mathbf{ite}(\texttt{home}[w,p] = j \wedge \texttt{away}[w,p] = i, 1, 0) \Big) = 1 \tag{24}$$

- **Period limit constraint:**

$$\forall t, p: \quad \sum_{w=1}^{n-1} (\mathbf{ite}(\texttt{home}[w,p] = t, 1, 0) + \mathbf{ite}(\texttt{away}[w,p] = t, 1, 0)) \leq 2 \tag{25}$$

Auxiliary Variable Definitions

$$\forall t: \quad \texttt{home\_count}[t] = \sum_{w=1}^{n-1} \sum_{p=1}^{n/2} \mathbf{ite}(\texttt{home}[w,p] = t, 1, 0) \tag{26}$$

$$\forall t: \quad \texttt{away\_count}[t] = \sum_{w=1}^{n-1} \sum_{p=1}^{n/2} \mathbf{ite}(\texttt{away}[w,p] = t, 1, 0) \tag{27}$$

Implied Constraints

- **Total matches per team:**

$$\forall t: \quad \texttt{home\_count}[t] + \texttt{away\_count}[t] = n - 1 \tag{28}$$

- **Total period appearances:**

$$\forall t: \quad \sum_{p=1}^{n/2} \sum_{w=1}^{n-1} (\mathbf{ite}(\texttt{home}[w,p] = t, 1, 0) + \mathbf{ite}(\texttt{away}[w,p] = t, 1, 0)) = n - 1 \tag{29}$$

Symmetry Breaking Constraints

- **Team symmetry:**

$$\forall p: \quad \texttt{home}[1,p] = 2p - 1 \tag{30}$$

$$\forall p: \quad \texttt{away}[1,p] = 2p \tag{31}$$

- **Lexicographic ordering for weeks:**

$$\forall w: \quad (\texttt{home}[w,1], \ldots, \texttt{home}[w,n/2], \texttt{away}[w,1], \ldots, \texttt{away}[w,n/2]) <_{\textbf{lex}} (\texttt{home}[w+1,1], \ldots, \texttt{home}[w+1,n/2], \texttt{away}[w+1,1], \ldots, \texttt{away}[w+1,n/2]) \tag{32}$$

- **Lexicographic ordering for periods:**

$$\forall p: \quad (\texttt{home}[1,p], \ldots, \texttt{home}[n-1,p], \texttt{away}[1,p], \ldots, \texttt{away}[n-1,p]) <_{\textbf{lex}} (\texttt{home}[1,p+1], \ldots, \texttt{home}[n-1,p+1], \texttt{away}[1,p+1], \ldots, \texttt{away}[n-1,p+1]) \tag{33}$$

4.4 Validation

The SMT model was implemented using both Z3 and CVC5 solvers to ensure robustness and enable solver comparison. To enable a solver-independent comparison, the model was also encoded in SMT-LIB format.

Experimental Results The SMT model was validated with both Z3 and CVC5 for the decision version and Z3 only for the optimization one:

ID	Z3 (all)	Z3 w/o SB	Z3 w/o IC	CVC5 (all)	CVC5 w/o SB	CVC5 w/o IC
6	0 | **1**	0 | **1**	0 | **1**	0 |	1 |	0 |
8	0 | **1**	2 | **1**	0 | **1**	18 |	33 |	20 |
10	30 | N/A	49 | N/A	37 | N/A	N/A |	N/A |	N/A |
12	216 | N/A	289 | N/A	N/A | N/A	N/A |	N/A |	N/A |

Table 4: Results using Z3 and CVC5 with the possible combinations of active constraints.

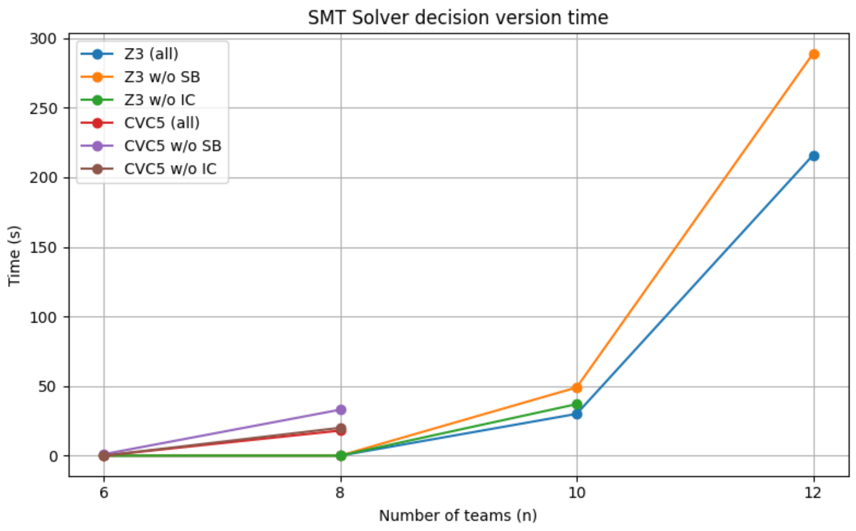

Figure 3: SMT solvers: solving time for different constraint configurations.

5 MIP Model

5.1 Decision Variables

The MIP model employs binary decision variables for precise match representation:

$$x_{w,p,i,j} \in \{0,1\} \quad \forall w,p,i,j,i \neq j \tag{34}$$

where $x_{w,p,i,j} = 1$ if team i plays at home against team j in week w, period p, and $x_{w,p,i,j} = 0$ otherwise. The use of binary variables, instead of integer variables as in the CP model, has proven to be significantly less time consuming.

Additionally, auxiliary variables are defined to facilitate the linear formulation of the objective function:

$$h_t \in \{0, 1, \ldots, n-1\} \quad \forall t \tag{35}$$
$$a_t \in \{0, 1, \ldots, n-1\} \quad \forall t \tag{36}$$
$$M \in \{0, 1, \ldots, n-1\} \tag{37}$$

where h_t represents the number of home games for team t, a_t represents the number of away games for team t, and M represents the maximum home/away imbalance.

5.2 Objective Function

The objective is to minimize the maximum imbalance in home and away games for any team. Since the absolute value function is non-linear, we linearize it using auxiliary variables:

$$\text{minimize} \quad M \tag{38}$$

subject to the linearization constraints:

$$h_t - a_t \leq M \quad \forall t \tag{39}$$
$$a_t - h_t \leq M \quad \forall t \tag{40}$$

This linear formulation ensures that $M \geq |h_t - a_t|$ for all teams t, and the minimization objective drives M to equal $\max_t |h_t - a_t|$.

5.3 Constraints

All constraints in the MIP model are linear and correspond to the common formalization described in Section 1.

Main Problem Constraints

- **Each slot has exactly one match:**

$$\sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} x_{w,p,i,j} = 1 \quad \forall w, p \tag{41}$$

- **Mutual exclusion constraint:** For each time slot, at most one of the two possible home/-away arrangements between any pair of teams can occur:

$$x_{w,p,i,j} + x_{w,p,j,i} \leq 1 \quad \forall w, p, i, j, i < j \tag{42}$$

- **Each pair of teams plays exactly once:**

$$\sum_{w=1}^{n-1} \sum_{p=1}^{n/2} (x_{w,p,i,j} + x_{w,p,j,i}) = 1 \quad \forall i, j, i < j \tag{43}$$

- **Each team plays exactly once per week:**

$$\sum_{p=1}^{n/2}\sum_{\substack{j=1\\j\neq i}}^{n}(x_{w,p,t,j}+x_{w,p,j,t})=1 \quad \forall w,t \tag{44}$$

- **Each team appears in the same period at most twice:**

$$\sum_{w=1}^{n-1}\sum_{\substack{j=1\\j\neq i}}^{n}(x_{w,p,t,j}+x_{w,p,j,t})\leq 2 \quad \forall p,t \tag{45}$$

Auxiliary Variable Definitions The home and away game counts are defined using linear constraints:

$$h_t=\sum_{w=1}^{n-1}\sum_{p=1}^{n/2}\sum_{\substack{j=1\\j\neq i}}^{n}x_{w,p,t,j} \quad \forall t \tag{46}$$

$$a_t=\sum_{w=1}^{n-1}\sum_{p=1}^{n/2}\sum_{\substack{j=1\\j\neq i}}^{n}x_{w,p,j,t} \quad \forall t \tag{47}$$

Implied Constraints

- **Total matches per team:**

$$h_t+a_t=n-1 \quad \forall t \tag{48}$$

Symmetry Breaking Constraints

- **Team symmetry:** Fix the schedule of the first week to break team symmetries:

$$x_{1,p,2p-1,2p}=1 \quad \forall p \tag{49}$$

$$x_{1,p,i,j}=0 \quad \forall p,(i,j)\neq(2p-1,2p), i\neq j \tag{50}$$

- **Team 1 opponent ordering constraint:** It fixes the order of team 1's opponents by ensuring that in week w, team 1 plays against team $w+1$:

$$\sum_{p=1}^{n/2}(x_{w,p,1,w+1}+x_{w,p,w+1,1})=1 \quad \forall w \tag{51}$$

5.4 Validation

The MIP model was implemented using PuLP, a Python library for linear programming that provides a solver-independent interface. This design choice allows the same model to be tested with different MIP solvers, in our case, CBC and GLPK.

Experimental Results The following tables and plot show the results of the two MIP solvers.

ID	CBC (all)	CBC w/o SB	CBC w/o IC	GLPK (all)	GLPK w/o SB	GLPK w/o IC
6	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**	0 | **1**
8	0 | **1**	0 | **1**	0 | **1**	2 | **1**	0 | **1**	1 | **1**
10	1 | **1**	2 | **1**	2 | **1**	1 | N/A	38 | N/A	59 | **1**
12	15 | N/A	17 | N/A	3 | 5	40 | N/A	74 | N/A	218 | N/A
14	29 | N/A	N/A | N/A	27 | N/A	N/A | N/A	N/A | N/A	N/A | N/A

Table 5: Results using CBC and GLPK with the possible combinations of active constraints.

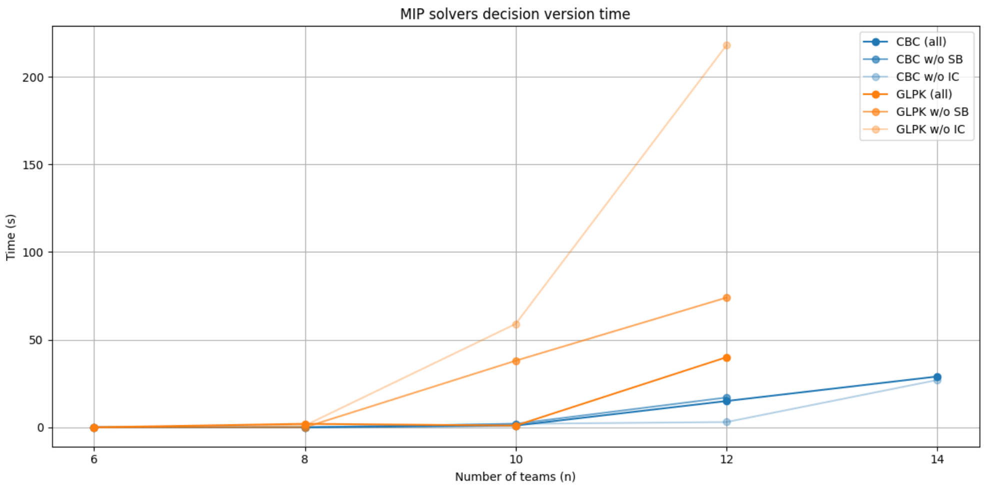

Figure 4: MIP solvers: solving time for different constraint configurations.

6 Conclusions

This comparative study reveals several important insights about multi-paradigm optimization for the Sports Tournament Scheduling problem:

1. **Optional constraints impact**: The added symmetry breaking and implied constraints can be for some model beneficial, some of them having great improvements particularly noticeable with larger number of teams.

2. **Encoding sensitivity**: SAT performance varies significantly with encoding choice, with bitwise encoding consistently outperforming alternatives.

3. **Scalability patterns**: CP excels in optimization objectives and modeling flexibility.

Authenticity and Author Contribution Statement

We declare that the work described in this report is our own and has not been copied from any other source. All external ideas, resources, or materials that have been used have been appropriately cited and referenced in the bibliography.

AI-generated content has been used in this report and it was limited to language and formulae refinement, conceptual clarification, without compromising academic integrity.

Author Contributions: Both authors contributed to every model, dividing them into GitHub issues as it is showed in `https://github.com/leo-tasso/STS/issues?q=is%3Aissue`.

References

[1] A Sport Scheduling problem example by *csplib*.
`https://www.csplib.org/Problems/prob026/models/SportsScheduling.py.html`

[2] *G. S. Tseitin*, "On the complexity of derivation in propositional calculus," Studies in Constructive Mathematics and Mathematical Logic, Part II, pp. 115–125, 1968.