CDMO Project: STS Problem

Gianlorenzo Urbano (gianlorenzo.urbano@studio.unibo.it)
Luca Lucioli (luca.lucioli@studio.unibo.it)
Michelangelo Urbano (michelangelo.urbano@studio.unibo.it)

November 30, 2025

1 Introduction

In this project, we address a specific variant of the Sports Tournament Scheduling (STS) problem by employing a decomposition strategy. Instead of solving the full problem monolithically, we utilize the *Circle Method* (canonical 1-factorization) as a pre-processing step to statically fix the weekly schedule. This approach reduces the combinatorial complexity of the decision and optimization tasks for our solvers (CP, SAT, MIP, SMT) to two specific sub-problems: assigning the pre-determined matches to periods and determining the home/away status. The following sections detail the mathematical formalization common to all our models, the algorithmic details of the pre-processing, and the theoretical properties inherent to the problem structure.

1.1 Common Formalization

The following formalization is shared across all approaches.

1.1.1 Input Parameters:

- Number of teams n (even) and the set of teams $\mathcal{T} = \{1, \dots, n\}$.
- Number of weeks $n_{weeks} = n - 1$ and the set of weeks $\mathcal{W} = \{1, \dots, n-1\}$.
- Number of periods $n_{periods} = n/2$ and the set of periods $\mathcal{P} = \{1, \dots, n/2\}$.
- The pre-computed match-week matrix W, where $W_{ij} \in \mathcal{W}$ denotes the week in which team i plays team j.

1.1.2 Decision Variables:

- $\pi_{ij} \in \mathcal{P}$: The period assigned to the match between teams i and j $(i < j)$.
- $h_{ij} \in \{0, 1\}$: Boolean variable, 1 if team i plays Home against j, 0 otherwise $(i < j)$.

1.1.3 Objective Function and Bounds

To minimize the disparity between home and away games, we introduce the objective variable β representing the *Total Imbalance.* For each team $t \in \mathcal{T}$, we define:

- $H_t = \sum_{j \in \mathcal{T}, j \neq t} h_{tj}$: The total number of home games played by team t.
- $A_t = \sum_{j \in \mathcal{T}, j \neq t} (1 - h_{tj})$: The total number of away games played by team t.
- $\delta_t = |H_t - A_t|$: The individual imbalance for team t.

The optimization goal is to minimize the maximum individual imbalance:

$$\text{minimize } \beta \quad \text{subject to} \quad \beta = \max_{t \in \mathcal{T}} \delta_t$$

Bounds: Since the total number of games per team is $n - 1$ (which is odd given that n is even), perfect balance ($\delta_t = 0$) is impossible. The minimum possible difference is 1 (e.g., $\frac{n}{2}$ home vs $\frac{n}{2} - 1$ away). The maximum possible difference occurs if a team plays all games at home or away ($n - 1$). Therefore, the domain of the objective variable is:

$$\beta \in [1, n - 1]$$

1.2 Pre-processing: The Circle Method.

Let n be the even number of teams. We designate team n as the fixed *pivot.* The remaining $n - 1$ teams form the "circle", represented formally as an ordered sequence $C = (c_0, c_1, \ldots, c_{n-2})$ where $c_i \in \{1, \ldots, n-1\}$. To analyze the impact of the initial configuration, we implemented two variants for C:

1. Standard: $C = (1, 2, \ldots, n - 1)$.
2. Rotated: $C = (2, 3, \ldots, n - 1, 1)$, obtained by shifting the first element to the end.

The schedule consists of $n - 1$ weeks, indexed by $w \in \{0, \ldots, n - 2\}$. For each week w, the set of matches is generated using modular arithmetic on the indices of C:

- The pivot plays against the team at index w: $\{n, c_w\}$.
- For every offset $k \in \{1, \ldots, \frac{n}{2} - 1\}$, the team at index $(w + k) \pmod{n - 1}$ plays against the team at index $(w - k) \pmod{n - 1}$.

The aggregation of these matches for all w yields the fixed weekly schedule, formally denoted hereafter as the matrix W (where W_{ij} indicates the week in which team i plays team j). This construction strictly guarantees the satisfaction of the following constraints:

- Every team plays with every other team exactly once.
- Every team plays exactly one match per week.

[1].

1.3 Implicit Properties and Symmetries

We identify specific theoretical properties and symmetries inherent to the problem structure, as analyzed in the literature [2].

Implicit Properties. Is identified a structural invariant common to all solutions of this problem, referred to as the *Deficient Teams* property. Theoretical analysis shows that in any valid schedule, for every period p, the set of teams partitions into two distinct subsets:

- A set $\mathcal{D}_p$ consisting of exactly two teams (called "deficient") that appear exactly once in period p.
- The remaining $n - 2$ teams, which appear exactly twice in period p.
- Furthermore, for any two distinct periods p and p', the deficient teams of p appear twice in p' (implying $\mathcal{D}_p \cap \mathcal{D}_{p'} = \emptyset$).

This property is intrinsic to the combinatorial structure of the problem and can be exploited later to define implicit constraints.

Symmetries This problem is intrinsically symmetric in fact equivalent solutions can typically be obtained through several transformations:

- Team Permutation: Renumbering the teams (e.g., swapping Team 1 and Team 2) results in an equivalent solution.
- Week Permutation: Swapping the entire schedule of two distinct weeks preserves the validity of the tournament.
- Period Permutation: The periods are indistinguishable; swapping the matches of period p_i with period p_j creates an equivalent valid schedule.

However, in our specific approach, the pre-processing phase (Circle Method) produces a fixed match-week matrix W. This constant input inherently breaks the Team and Week permutation symmetries, as the specific pairings for every week are pre-determined.

2 CP Model

2.1 Decision variables

We adopt the decision variables π_{ij} and h_{ij} formally defined in the Common Formalization (see Section 1). However, specific to the CP implementation, the domain of the period variable is extended to $\mathcal{P} \cup \{0\}$. Semantically, the value 0 is reserved to represent the concept of a "null period" (applicable to diagonal entries where $i = j$), while values in $\mathcal{P}$ denote valid tournament periods.

2.2 Objective function

The optimization goal is to minimize the *Total Imbalance* β. The auxiliary variables required to compute it (H_t, A_t, and δ_t) are implemented exactly as defined in the Common Formalization (see Section 1). Furthermore, the bounds derived from the parity properties discussed in Section 1 (i.e., that a perfect balance of 0 is impossible for an even number of teams) are enforced as domain constraints:

$$1 \leq \beta \leq n - 1$$

2.3 Constraints

The CP formulation leverages the pre-computed match-week matrix W (defined in the Common Formalization, Section 1) as a fixed input parameter. The constraints are organized into three categories: main constraints, which ensure the strict validity of the schedule; implied constraints, introduced to improve propagation efficiency; and symmetry-breaking constraints, employed to eliminate symmetric equivalent solutions.

2.3.1 Main Constraints

The following constraints are strictly necessary to define a valid solution to the STS problem.

Matrix Properties (Validity and Symmetry). The variable matrix π must maintain a symmetric structure to reflect the undirected nature of the matches. Since the match $\{i, j\}$ is identical to the match $\{j, i\}$, both entries must be assigned to the same valid period in $\mathcal{P}$, while self-assignments are null:

$$\forall i, j \in \mathcal{T}, i \neq j : \quad \pi_{ij} \in \mathcal{P} \quad \wedge \quad \pi_{ij} = \pi_{ji}$$

$$\forall i \in \mathcal{T} : \quad \pi_{ii} = 0$$

Implementation: This is enforced by imposing inequality constraints ($\pi_{ij} \neq 0$) for off-diagonal entries and equality constraints ($\pi_{ij} = \pi_{ji}$) for all $i < j$ to guarantee the symmetric structure of the decision variables.

Home/Away Consistency. The home/away roles must be reciprocal. If team i plays Home against j, team j must necessarily play Away against i:

$$\forall i, j \in \mathcal{T}, i < j : \quad h_{ji} = 1 - h_{ij}$$

Implementation: The implementation follows the definition directly. We enforce a strict arithmetic equality between the symmetric entries of the Boolean matrix h, ensuring that h_{ji} is always the complement of h_{ij}.

Period Usage. Each team can play at most twice within the same period p over the tournament. Mathematically:

$$\forall t \in \mathcal{T}, \forall p \in \mathcal{P} : \quad \sum_{j \in \mathcal{T}, j \neq t} \mathbb{I}(\pi_{tj} = p) \leq 2$$

Implementation: This requirement is enforced using the global constraint `count_geq`(x, y, c), which constrains the parameter c to be greater than or equal to the number of occurrences of value y in array x. By setting $c = 2$, we effectively impose that $2 \geq$ occurrences, which satisfies the "at most 2" requirement. Crucially, we annotated this constraint with `domain_propagation` to enforce *Generalized Arc Consistency (GAC)*. Experimental testing revealed that this specific formulation yielded better solving times than alternative global constraints such as `global_cardinality_closed`.

Weekly Structure. The pre-processing phase (Circle Method) assigns each match $\{i, j\}$ to a specific week W_{ij}. We must ensure that for every week w and every period p, exactly one match is scheduled:

$$\forall w \in \mathcal{W}, \forall p \in \mathcal{P} : \quad \sum_{i<j,\, W_{ij}=w} \mathbb{I}(\pi_{ij} = p) = 1$$

Implementation: The implementation is a direct translation of the formulation above. We simply sum the Boolean indicators for the specific subset of matches belonging to week w (i.e., those where $W_{ij} = w$) and constrain the result to be exactly 1. A variant using the `alldifferent` global constraint on the period variables of each week was also tested but resulted in lower performance compared to this summation formulation.

2.3.2 Implied constraints

To enhance propagation efficiency, we leverage the structural properties derived from the *Deficient Teams* invariant previously discussed in Section 1.3 [2]. Although redundant for correctness, enforcing these properties narrows the search space significantly.

Deficient Teams Structure. As established, for every period p, exactly two teams must play exactly once (deficient), while the remaining $n-2$ teams must play exactly twice. We enforce these two conditions mathematically as follows:

$$\forall p \in \mathcal{P} : \quad \sum_{t \in \mathcal{T}} \mathbb{I}\left(\sum_{j \neq t} \mathbb{I}(\pi_{tj} = p) = 1 \right) = 2 \tag{1}$$

$$\forall p \in \mathcal{P} : \quad \sum_{t \in \mathcal{T}} \mathbb{I}\left(\sum_{j \neq t} \mathbb{I}(\pi_{tj} = p) = 2 \right) = n - 2 \tag{2}$$

Implementation: We implemented equations (1) and (2) using nested Boolean indicators evaluated directly on the decision variables π_{tj}, without reifying the intermediate occurrence counts into auxiliary integer variables. This formulation creates a direct propagation path from the period assignments to the constraints, which was empirically observed to be significantly faster than alternatives based on `global_cardinality_closed` or `count_eq`.

Unique Deficiency per Team. Furthermore, theoretical analysis implies that each team is deficient in exactly one period across the entire tournament. In our CP model, we enforce a relaxed version of this property:

$$\forall t \in \mathcal{T} : \quad \sum_{p \in \mathcal{P}} \mathbb{I}(\text{is\_deficient}(t, p)) \leq 1 \tag{3}$$

where the Boolean auxiliary variable is_deficient(t, p) is formally defined by the equivalence:

$$\text{is\_deficient}(t,p) \iff \left(\sum_{j \in \mathcal{T}, j \neq t} \mathbb{I}(\pi_{tj} = p) = 1 \right)$$

Implementation: We enforce this using the global constraint `count_leq`(x, y, c), which imposes the condition $c \leq |\{k \mid x_k = y\}|$. By mapping x to the team's deficiency indicators ($[\text{is\_deficient}_{tp} \mid p \in \mathcal{P}]$), setting the target value $y =$ `true`, and the threshold $c = 1$, we effectively constrain each team to be deficient in at least one period (i.e., $\sum_p \mathbb{I}(\text{is\_deficient}_{tp}) \geq 1$). Although the strict theoretical requirement is "exactly 1", enforcing the weaker "at least 1" condition proved computationally more efficient while still providing sufficient filtering power. The strict "exactly 1" condition is implicitly satisfied when combined with the other model constraints. However, in the optimization models, where strong propagation is critical to prune suboptimal branches, we explicitly enforce the upper bound as well using a semantically redundant constraint:

$$\sum_{p \in \mathcal{P}} \mathbb{I}(\text{is\_deficient}_{tp}) \leq 1$$

This creates a hybrid formulation where the global constraint enforces the lower bound and the linear constraint enforces the upper bound. Experimental evidence confirmed that this combination significantly improves solving efficiency compared to using either constraint in isolation, likely by enabling stronger domain pruning through complementary propagators.

2.3.3 Symmetry breaking constraints

As discussed in Section 1.3 [2], the problem exhibits specific symmetries that survived the pre-processing phase. We introduce additional constraints to eliminate equivalent solutions and select a unique canonical representative for each symmetry class.

Period Ordering. Since the period labels $p \in \mathcal{P}$ are arbitrary and indistinguishable, simply renumbering the periods results in an equivalent solution. To break this symmetry, we impose a strict ordering on the periods assigned to the matches of the first week ($w = 1$). Let S_1 be the sequence of period variables corresponding to the matches scheduled in week 1, ordered by team index:

$$S_1 = \langle \pi_{ij} \mid i < j,\, W_{ij} = 1 \rangle$$

Implementation: We enforce that the sequence S_1 must be non-decreasing using the global constraint `increasing`(S_1). This effectively forces the solver to explore only one permutation of the periods for the first week (e.g., mapping the first match to period 1, the second to period 2, etc., as much as satisfying validity allows). Experimental results indicated that the `increasing` constraint provides a better trade-off between propagation cost and search space reduction compared to variants like `strictly_increasing` or `lex_lesseq`.

Home/Away Symmetry. The strategy to break the home/away reflection symmetry differs depending on the problem version:

1. *Decision Version:* Since no objective function is present, the home/away balance is irrelevant. We can arbitrarily fix the entire home/away structure to remove all symmetries and variables. We enforce that the team with the lower index always plays Home:

$$\forall i, j \in \mathcal{T}, i < j : \quad h_{ij} = 1$$

This trivializes the search for h variables, leaving the solver to focus solely on period assignments.

2. *Optimization Version:* We cannot fix all roles because the imbalance objective β depends on the specific distribution of home games. However, we can still break the global "mirror" symmetry (swapping Home/Away for everyone) by fixing the schedule of a single pivot team (Team 1). We force Team 1 to play Home for the first half of its opponents and Away for the second half:

$$\forall j \in \mathcal{T} \setminus \{1\} : \quad h_{1j} = \begin{cases} 1 & \text{if } j \leq n/2 \\ 0 & \text{if } j > n/2 \end{cases}$$

Implementation: These constraints are implemented as direct equality assignments on the h_{ij} variables.

2.4 Validation

The CP model was implemented in MiniZinc and executed using the Python `minizinc` library to automate the generation of instances and the collection of results.

2.4.1 Experimental Design

To ensure the reproducibility and reliability of our results, the experimental study was conducted according to the following setup:

- Hardware and Software Environment: All experiments were performed inside a Docker container to guarantee a consistent execution environment. The host machine was a MacBook M1 PRO. The software stack included MiniZinc 2.9.4 and Python 3.12.
- Timeout: A strict time limit of 300 seconds (5 minutes) was imposed for each run. If a solver failed to find a solution (or prove optimality) within this limit, the execution was terminated, and the best solution found so far (if any) was recorded.
- Input Generation: The instance data (specifically the week matrix W) was generated dynamically using the Circle Method algorithm implemented in Python and passed to the MiniZinc model via a temporary `.dzn` file.
- Reproducibility: To ensure deterministic behavior, the random seed for the solvers was fixed to 42.

We evaluated the performance of our model using two distinct solvers: Gecode (standard finite domain solver) and Chuffed (lazy clause generation solver). To assess the impact of our modeling choices, we defined three incremental model configurations:

1. Base: Includes only the Main Constraints (Section 2.3.1).

2. Base_SB: Adds the Symmetry Breaking constraints (Section 2.3.2).
3. Base_SB_implicit: The complete model, including Implied constraints (Section 2.3.3).

For the Decision Problem, we experimented with the following search strategies on the complete model:

- Default solver search strategy.
- ff (*First-Fail* heuristic): Select the variable with the smallest domain size and assigning the minimum value first.
- ff_geometric: We apply a *Geometric restart* policy (scale factor 1.5, base 100) combined with the *First-Fail* variable selection and indomain_min value selection.
- ff_luby: We apply a *Luby restart* sequence (scale factor 100) with the*First-Fail* variable selection and indomain_min value selection.

For the Optimization Problem, we tested:

- luby: A complete search using *First-Fail* on all variables combined with *Luby restarts* (scale factor 50).
- rr: A *Large Neighborhood Search* (LNS) approach. We utilize the *Relax-and-Reconstruct* meta-heuristic (fixing 85% of variables and re-solving). The variable selection relies on *dom/wdeg* (domain size divided by weighted degree) to prioritize hard-to-satisfy constraints.

For the sake of brevity, we omit detailed results for other solver configurations tested during the preliminary phase that yielded inferior performance. Regarding search heuristics, we evaluated the input_order variable selection (lexicographic) and indomain_random value selection; however, both strategies resulted in significantly larger search trees compared to the first_fail principle, failing to converge on valid solutions for instances with $n > 14$.

2.4.2 Experimental results

The results are reported in the following tables. We evaluate the progressive impact of our modeling choices (*Base* $\rightarrow$ *Base_SB* $\rightarrow$ *Base_SB_implicit*) and the effectiveness of the search strategies defined in the previous section.

Decision Problem. Table 1 details the runtime (in seconds) for all tested configurations. We compare the default search strategy on the incremental models against the specific strategies (ff, ff_geometric, ff_luby) applied to the complete model (*Base_SB_implicit*).

Optimization Problem. Table 2 reports the best objective value (β) found within the 300s time limit. We compare the default search on the incremental models against the *luby* and *rr* strategies on the complete model.

	Gecode						Chuffed					
ID	Base	Base_SB	Imp	ff	ff_geo	ff_luby	Base	Base_SB	Imp	ff	ff_geo	ff_luby
04	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
06	0	0	0	0	0	0	0	0	0	0	0	0
08	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	1	2	0	0	0	0	0	0
12	0	0	0	0	2	11	0	0	0	0	0	0
14	132	6	56	75	259	-	12	0	0	1	1	2
16	-	-	-	-	-	-	7	13	26	6	8	7
18	-	-	-	-	-	-	-	-	4	31	29	30
20	-	-	-	-	-	-	-	-	14	-	-	-

Table 1: Runtime (s) for the Decision Problem. Columns "Imp" refer to Base_SB_implicit with default search. Columns "ff", "ff_geo", "ff_luby" refer to Base_SB_implicit with the respective strategy.

	Gecode					Chuffed				
ID	Base	Base_SB	Imp	luby	rr	Base	Base_SB	Imp	luby	rr
04	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
06	1	1	1	1	1	1	1	1	1	1
08	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
12	3	1	1	1	1	1	1	1	1	1
14	3	1	1	-	1	1	1	1	-	1
16	-	-	-	-	-	-	1	1	-	1
18	-	-	-	-	-	-	11	1	-	1
20	-	-	-	-	-	-	-	-	-	-

Table 2: Best Total Imbalance (β) found. Columns "luby" and "rr" use the complete model Base_SB_implicit.

Discussion of Results. The experimental evidence highlights a significant performance gap between the two solvers, with Chuffed strictly outperforming Gecode across both decision and optimization tasks.

Regarding the Decision Problem (Table 1):

- Solver Performance: Chuffed demonstrates superior scalability, solving instances up to $n = 20$ (with Implied constraints), whereas Gecode fails to converge for any instance larger than $n = 14$.
- Impact of Constraints: Symmetry Breaking (SB) is crucial for performance. In Gecode ($n = 14$), adding SB reduces runtime from 132s to 6s. The Implied constraints (Imp) prove essential for the hardest solved instance: Chuffed solves $n = 20$ in 14s only with the Implied model, failing ("-") with lighter configurations.
- Search Strategies: The default solver search strategy generally outperforms the manual first_fail heuristic and restart-based strategies (geo, luby). This is evident in Chuffed for $n = 20$, where the default search converges in 14s, while the other search strategies fail to find a solution within the time limit.

Regarding the Optimization Problem (Table 2):

- Optimality: Chuffed proves optimality up to $n = 18$, while Gecode is limited to $n = 14$.
- Strategies: The Large Neighborhood Search strategy (`rr`) proved highly effective, enabling Chuffed to prove optimality for instances up to $n = 18$. In stark contrast, the complete search with Luby restarts (`luby`) exhibited significantly poorer scalability: it failed to prove optimality for $n \geq 14$.

3 SAT Model

3.1 Overview and Strategy

Consistent with the project-wide decomposition, we use the *Circle Method* (Section 1.2) to fix the weekly matches, yielding the matrix W. The SAT model therefore decides only: (i) the *period assignment* of each pre-determined match; and (ii) its *home/away* orientation. Structural constraints are encoded with Boolean variables and cardinality encodings.

3.2 Decision variables

Given teams $\mathcal{T} = \{1, \dots, n\}$, weeks $\mathcal{W} = \{1, \dots, n-1\}$, and periods $\mathcal{P} = \{1, \dots, n/2\}$, we introduce:

- $mp_{i,j,p}$: Boolean channel of the integer period variable π_{ij} (Section 1). For $i < j$ and $p \in \mathcal{P}$, $mp_{i,j,p} = \texttt{true}$ iff match $\{i, j\}$ (whose week $w = W_{ij}$ is fixed by preprocessing) is played in period p.
- $h_{i,j}$: for $i < j$, $h_{i,j} = \texttt{true}$ iff team i plays Home against j (thus $h_{j,i} = \neg h_{i,j}$ by reciprocity).

To exploit the *Deficient Teams* structure (Section 1.3), we also use:

- $def_{t,p}$: `true` iff team t is *deficient* in period p, i.e., it plays exactly one match in period p over the whole tournament.

3.3 Objective function

Optimization is handled by a binary search on the imbalance variable β (Section 1). For a fixed candidate bound β, each team t must satisfy:

$$\left\lceil \tfrac{(n-1)-\beta}{2} \right\rceil \;\le\; \sum_{j \neq t} \mathbb{I}(t \text{ plays Home vs } j) \;\le\; \left\lfloor \tfrac{(n-1)+\beta}{2} \right\rfloor.$$

We iteratively solve SAT decision problems, tightening the bound until minimal β is found.

3.4 Constraints

Main constraints.

1. **Unique period assignment:** each match $\{i, j\}$ is assigned to exactly one period:
$$\sum_{p \in \mathcal{P}} mp_{i,j,p} = 1.$$
2. **Slot capacity:** for every week w and period p, exactly one match of week w occupies period p:
$$\sum_{\{i,j\} \in M_w} mp_{i,j,p} = 1 \quad (w \in \mathcal{W},\ p \in \mathcal{P}).$$
3. **Per-team per-period cap:** each team appears in at most 2 matches in the same period across the season:
$$\sum_{j \neq t} mp_{t,j,p} \le 2 \quad (t \in \mathcal{T},\ p \in \mathcal{P}).$$

Implied constraints (Deficient Teams). Using $def_{t,p}$ to capture deficiency:

1. **Definition:**
$$def_{t,p} \iff \Big(\sum_{j\neq t} mp_{t,j,p} = 1\Big).$$
Implemented by implications enforcing at-least-one and not-at-least-two; at-least-three is excluded by the main cap.

2. **Exactly two deficient per period:**
$$\sum_{t\in\mathcal{T}} def_{t,p} = 2 \quad (p \in \mathcal{P}).$$
Encoded using the *Sequential Counter* encoding.

3. **At most one deficient period per team:**
$$\sum_{p\in\mathcal{P}} def_{t,p} \leq 1 \quad (t \in \mathcal{T}).$$
Encoded using *Pairwise* at-most-one clauses.

Symmetry breaking. We reduce residual symmetries via:

1. **Home/Away symmetry:** fix Team 1's orientation pattern along weeks: it plays Home in the first $\lfloor (n-1)/2 \rfloor$ matches and Away thereafter (ordered by week), anchoring the global flip.

2. **Period lexicographic ordering:** periods are interchangeable; we impose a lexicographic order on the vectors $\text{Vector}(p) = \langle mp_{i,j,p} \rangle_{\{i,j\}\in M_1\cup\cdots\cup M_{n-1}}$ so that $\text{Vector}(p) \leq_{lex} \text{Vector}(p+1)$ for $p = 1, \ldots, |\mathcal{P}| - 1$.

3.5 Validation

Experimental design

The SAT solver utilized is Z3 accessed via its Python API.

The validation strategy follows a two-phase approach: in the first phase (*Vanilla*) we performed a comparative analysis of four pure cardinality constraint encodings to identify the most robust baseline. The "Vanilla" solvers implement the standard constraints using *Pairwise* (VP in the table), *Heule* (VH), *Sequential Counter* (VS) and *Totalizer* (VT). Using the most effective encoding logic derived from the previous phase (implemented in the "Smart" baseline), we evaluated the impact of domain-specific optimizations through four configurations:

- **Smart:** The baseline solver using hybrid efficient encodings (`smart_exactly_k`).
- **Smart + SB:** Adds Symmetry Breaking constraints (Home/Away fixing and Period Lexicographic ordering).
- **Smart + DT:** Adds the *Deficient Teams* implied constraints and auxiliary variables.
- **Smart + SB + DT:** Combines all techniques to assess their synergistic effect.

Experimental results

	Decisional								Optimization							
ID	**VP**	**VH**	**VS**	**VT**	**S**	**S_DT**	**S_SB**	**S_SB_DT**	**VP**	**VH**	**VS**	**VT**	**S**	**S_DT**	**S_SB**	**S_SB_DT**
12	1	1	0	0	0	0	0	0	**2**	**1**	**0**	**1**	**1**	**1**	**1**	**1**
14	2	2	0	2	1	1	1	1	**3**	**3**	**4**	**3**	**5**	**2**	**4**	**2**
16	9	5	4	8	10	10	19	10	**9**	**17**	**2**	**43**	**46**	**5**	**68**	**6**
18	26	14	15	29	31	34	1	5	**45**	**187**	**73**	**223**	**121**	**23**	N/A	**17**
20	48	272	94	230	73	104	169	42	N/A	300|10	N/A	N/A	300|5	**161**	N/A	**111**
22	N/A	N/A	N/A	N/A	N/A	80	N/A	N/A	N/A	N/A	N/A	N/A	N/A	300|5	N/A	**195**

Table 3: Experimental results for SAT decisional and optimization models. Time is reported in seconds. In the optimization columns, bold numbers indicate the time to prove the optimal solution. Entries in the format $Time|Obj$ indicate a timeout where the solver returned the best objective value found within the time limit.

The initial encoding comparisons revealed the unexpected robustness of the *Pairwise* encoding (VP), which outperformed more compact alternatives like Totalizer on instances up to $N = 20$, suggesting that their stronger unit propagation outweighs the overhead of the larger formula size. The transition to structural optimizations demonstrated that explicitly encoding the DT invariant is the most effective single factor for optimization, enabling the solver to prove optimality for $N = 20$ where the baseline failed. While SB alone incurred performance penalties on some instances, its synergy with DT yielded the best overall performance. A remarkable anomaly appears for $N = 22$ with the **Smart+SB+DT** configuration, where the solver successfully finds the optimal solution but fails to solve the decisional version. This could be explained by the constraining power of the tight objective bound ($\beta = 1$) combined with structural pruning that drastically reduces the search space, whereas the unconstrained decisional version leaves the solver navigating a vast and complex landscape without sufficient guidance.

4 SMT Model

4.1 Overview and Strategy

We adopt the same decomposition used throughout the project: the weekly schedule is fixed by the *Circle Method* (Section 1.2), yielding the match-week matrix W. The SMT model then solves only the *period assignment* and (optionally) the *home/away* orientation. This separation drastically reduces the search space and enables efficient pseudo-Boolean encodings.

4.2 Decision variables

Consistent with the Common Formalization (Section 1), we use:

- Period variables $\pi_{ij} \in \mathcal{P}$ for all $i < j$ with $W_{ij} \in \mathcal{W}$; and $\pi_{ii} = 0$.
- Home/Away variables $h_{ij} \in \{0, 1\}$ for all $i < j$, with reciprocal consistency $h_{ji} = 1 - h_{ij}$.

In SMT we encode period selections via guarded pseudo-Boolean literals of the form $[\pi_{ij} = p]$ for $p \in \mathcal{P}$.

4.3 Constraints

The model enforces the same semantic constraints as in CP, using pseudo-Boolean forms:

- **Weekly structure:** For every week w and every period p, exactly one match of week w is assigned to p:
$$\forall w \in \mathcal{W}, \forall p \in \mathcal{P} : \sum_{i<j,\, W_{ij}=w} \mathbb{I}(\pi_{ij} = p) = 1.$$

- **Period usage (per-team at-most-two):** For every team t and period p:
$$\sum_{j \in \mathcal{T},\, j \neq t} \mathbb{I}(\pi_{tj} = p) \leq 2.$$

- **Deficient teams structure (implied):** For every period p exactly two teams appear exactly once, and the remaining $n-2$ appear exactly twice:
$$\sum_{t \in \mathcal{T}} \mathbb{I}\Big(\sum_{j \neq t} \mathbb{I}(\pi_{tj} = p) = 1 \Big) = 2, \tag{4}$$
$$\sum_{t \in \mathcal{T}} \mathbb{I}\Big(\sum_{j \neq t} \mathbb{I}(\pi_{tj} = p) = 2 \Big) = n - 2. \tag{5}$$

- **Unique deficiency per team (implied/relaxed):** Each team is deficient in at most one period:
$$\forall t \in \mathcal{T} : \quad \sum_{p \in \mathcal{P}} \mathbb{I}\Big(\sum_{j \neq t} \mathbb{I}(\pi_{tj} = p) = 1 \Big) \leq 1.$$

- **Symmetry breaking (periods):** The periods in week $w = 1$ follow a non-decreasing order over the lexicographically ordered matches $\langle \pi_{ij} \mid i < j, W_{ij} = 1 \rangle$.

- **Home/Away symmetry:** As in CP, we fix a canonical orientation for the decision version (Section 1), preserving reciprocity $h_{ji} = 1 - h_{ij}$.

4.4 Encoding and Solver

Cardinality constraints are expressed with pseudo-Boolean operators (Exactly-One and At-Most) over the guards $[\pi_{ij} = p]$. We apply the Z3 tactic `card2bv` followed by `smt` to compile cardinalities into efficient bit-vector operations. For optimization we switch to `Optimize` with a minimization objective.

4.5 Objective function (optimization version)

We minimize the *Total Imbalance* β as in Section 1, by counting per-team H_t (home) and A_t (away) occurrences induced by h_{ij} and enforcing $\delta_t \geq |H_t - A_t|$ with standard linearization. We minimize the maximum imbalance via a variable *total_imbalance* such that *total_imbalance* $\geq \delta_t$ for all t, and then min *total_imbalance*.

4.6 Validation and Results

We evaluate the SMT variants (including *presolve_2* and the baseline).

SMT Decision

Model	n	Time (s)	Objective
baseline	4	0	-
baseline	6	0	-
baseline	8	0	-
baseline	10	9	-
baseline	12	183	-
baseline	14	300	-
compact	4	0	-
compact	6	0	-
compact	8	0	-
compact	10	0	-
compact	12	3	-
compact	14	4	-
compact	16	120	-
presolve	4	0	-
presolve	6	0	-
presolve	8	0	-
presolve	10	0	-
presolve	12	0	-
presolve	14	0	-
presolve	16	1	-
presolve	18	2	-
presolve	20	50	-
presolve	22	300	-
presolve	24	300	-
presolve_2	4	0	-
presolve_2	6	0	-
presolve_2	8	0	-
presolve_2	10	0	-
presolve_2	12	0	-
presolve_2	14	0	-
presolve_2	16	1	-
presolve_2	18	2	-
presolve_2	20	16	-
presolve_2	22	37	-
presolve_2	24	127	-
presolve_2	26	186	-
presolve_2	28	300	-

SMT Optimization

Model	n	Time (s)	Objective
presolve	4	0	-
presolve	6	0	1
presolve	8	0	1
presolve	10	0	1
presolve	12	0	1
presolve	14	1	1
presolve	16	9	1
presolve	18	17	1
presolve	20	176	1
presolve	22	300	-
presolve_2	4	0	-
presolve_2	6	0	1
presolve_2	8	0	1
presolve_2	10	0	1
presolve_2	12	0	1
presolve_2	14	1	1
presolve_2	16	9	1
presolve_2	18	18	1
presolve_2	20	300	-
presolve_2	22	300	-

5 MIP Model

5.1 Overview and Strategy

We mirror the project-wide decomposition: the weekly match matrix W is fixed by the *Circle Method* (Section 1.2). The MIP stage assigns periods to those predetermined matches, enforcing the per-team period usage limits and weekly structure. This reduces variable and constraint growth compared to direct MIP that jointly decides pairings and periods.

5.2 Decision variables

For each week $w \in \mathcal{W}$, period $p \in \mathcal{P}$, and match (i, j) such that $W_{ij} = w$ with $i < j$, we introduce binary assignment variables

$$x_{w,p,i,j} \in \{0, 1\} \quad \text{meaning match } \{i, j\} \text{ in week } w \text{ is scheduled in period } p.$$

Counts scale as $|x| = (n-1) \cdot |\mathcal{P}| \cdot |M_w| = \frac{n^2(n-1)}{4}$.

5.3 Constraints

All constraints are linear and adhere to the common semantics:

- **Match to one period:** For each (i, j) in week w,
$$\sum_{p \in \mathcal{P}} x_{w,p,i,j} = 1.$$
- **Exactly one match per week/period:** For each w and p,
$$\sum_{(i,j) \in M_w} x_{w,p,i,j} = 1.$$
- **Per-team at-most-two per period:** For each team t and period p,
$$\sum_{w \in \mathcal{W}} \sum_{\substack{(i,j) \in M_w \\ t \in \{i,j\}}} x_{w,p,i,j} \le 2.$$

To generate constraints efficiently, we precompute the index set team_matches$[t] = \{(w, i, j) \mid (i, j) \in M_w,\ t \in \{i, j\}\}$, yielding $O(n^2)$ model building rather than $O(n^3)$.

5.4 Objective function (optimization version)

We minimize the *Total Imbalance* β through standard linearization consistent with Section 1. Let

$$exthome(t) = \sum_{w \in \mathcal{W}} \sum_{p \in \mathcal{P}} \sum_{\substack{(i,j) \in M_w \\ i=t}} x_{w,p,i,j}, \tag{6}$$

$$extaway(t) = \sum_{w \in \mathcal{W}} \sum_{p \in \mathcal{P}} \sum_{\substack{(i,j) \in M_w \\ j=t}} x_{w,p,i,j}, \tag{7}$$

and introduce $\delta_t \geq |\text{home}(t) - \text{away}(t)|$ via

$$\delta_t \geq \text{home}(t) - \text{away}(t), \tag{8}$$

$$\delta_t \geq \text{away}(t) - \text{home}(t). \tag{9}$$

Finally, enforce $\mathit{total\_imbalance} \geq \delta_t$ for all t and minimize $\mathit{total\_imbalance}$.

5.5 Validation and Results

We compare our presolved MIP against a standard MIP formulation. Backends are OR-Tools compatible; default runs used SCIP with a 300s time limit, single thread.

MIP Decision

Model	n	Time (s)	Objective
presolve	4	0	-
presolve	6	0	-
presolve	8	0	-
presolve	10	0	-
presolve	12	1	-
presolve	14	4	-
presolve	16	11	-
presolve	18	115	-
presolve	20	41	-
presolve	22	300	-
standard	4	0	-
standard	6	0	-
standard	8	0	-
standard	10	0	-
standard	12	13	-
standard	14	300	-

MIP Optimization

Model	n	Time (s)	Objective
presolve	4	0	-
presolve	6	0	1
presolve	8	0	1
presolve	10	0	1
presolve	12	2	1
presolve	14	31	1
presolve	16	300	1
presolve	18	300	-
standard	4	0	-

Comparison with direct MIP.

n	Direct MIP	Presolve MIP	Speedup
12	45 s	1 s	45×
14	280 s	3 s	93×
16	timeout	15 s	>20×
18	timeout	45 s	>6×

These results show that fixing W and only assigning periods yields large speedups, smaller models, and near-optimal balance in the optimization variant.

6 Conclusions

These results show that fixing W and only assigning periods yields large speedups, smaller models, and near-optimal balance in the optimization variant.

In this work, we tackled the Sports Tournament Scheduling problem by comparing Constraint Programming, SAT, SMT, and Mixed-Integer Programming. The cornerstone of our approach was the decomposition strategy based on the Circle Method, which drastically reduced the combinatorial search space by statically fixing the weekly schedule. Experimental results highlighted that:

- In **CP**, the use of Lazy Clause Generation (Chuffed) combined with symmetry breaking proved superior to standard search.
- In **SAT** and **SMT**, the explicit encoding of the *Deficient Teams* invariant was the decisive factor in proving optimality for larger instances ($n \geq 20$).
- In **MIP**, the decomposition strategy was transformative, yielding speedups of over $90\times$ compared to the monolithic formulation.

Overall, the project demonstrates that hybridizing domain-specific theoretical properties with structural decomposition is more effective than relying solely on raw solver power. Ultimately, the SMT approach emerged as the best performing method, demonstrating superior scalability by solving the largest decision instances (up to $n = 26$) within the time limit, outperforming SAT, CP, and MIP.

Authenticity and Author Contribution Statement

Authenticity: This report and the accompanying source code are the original work of the authors named on the title page. Where external ideas, datasets, libraries, or prior results have been used, they are cited in the bibliography and acknowledged in the code or documentation.

Use of AI tools: We employed AI assistants (GitHub Copilot, GPT-5) to accelerate routine tasks such as refactoring Python scripts, drafting section outlines, formatting LaTeX tables, and generating shell scaffolding. All core modeling choices (formalization, constraints, encodings, and objectives), solver-specific implementations (CP, SAT, SMT, MIP), and experimental designs were conceived, implemented, and validated by the authors. **Author contributions:**

- Gianlorenzo Urbano: Common formalization; MIP and SMT implementations; presolve design and validation; integration of results into the report; experimental orchestration and reproducibility scripts.
- Luca Lucioli: Common formalization; SAT encodings (pairwise, totalizer, sequential counter, Heule); symmetry breaking design; DT implied constraints in SAT; consolidation of decisional/optimization results.
- Michelangelo Urbano: Common formalization; CP formulations and backend configuration; optimization objective linearization and analysis;.

References

[1] Wikipedia contributors, *Round-robin tournament*, Wikipedia, The Free Encyclopedia. Available at: `https://en.wikipedia.org/wiki/Round-robin_tournament` [Accessed: 15 June 2025].

[2] J.-P. Hamiez and J.-K. Hao, "Using solution properties within an enumerative search to solve a sports league scheduling problem," *Discrete Applied Mathematics*, vol. 156, pp. 1683–1693, 2008.