Combinatorial Decision Making And Optimization

Cirone Cono, cono.cirone@studio.unibo.it
Petrozziello Giulio, giulio.petrozziello@studio.unibo.it
Dardini Jacopo, jacopo.dardini@studio.unibo.it
Formichella Gio, gio.formichella@studio.unibo.it

1 Introduction

In this project, we address the sports tournament scheduling (STS) problem by applying a unified modeling approach, inspired by [1], across four computational paradigms: Constraint Programming (CP), Boolean Satisfiability (SAT), Satisfiability Modulo Theory (SMT) and Mixed Integer Programming (MIP). We begin by considering the structure of a round-robin tournament: for N teams, the tournament spans $N-1$ weeks, with each week consisting of $\frac{N}{2}$ games, one for each period. This structure naturally corresponds to a 1-factorization of the complete graph K_N on N vertices [2]. Given this structure, all four approaches begin by precomputing the round-robin tournament matrix using a standard 1-factorization technique, yielding a $\frac{N}{2} \times (N-1)$ matrix where each column corresponds to a week and contains the set of weekly matchups. Our experimental results show that the time required to precompute the underlying round-robin schedule is negligible (less than a second) for all considered instance sizes. Precomputing the matrix provides several advantages:

- It removes symmetries across weeks and matchup pairings, reducing the search space.
- It guarantees that each team plays exactly once per week.
- It ensures that each team plays against each other only once.

A balanced tournament design is a permutation of the order of games within each week that limits the number of times each team plays in the same period, as described in [3]. While the round-robin matrix defines the matchups, our models aim to compute an optimally balanced schedule, minimizing the difference of home and away games for each team, by assigning games to periods and slots in a way that satisfies the balancing constraints. All tests were run on a single core of an Intel i9-10900KF CPU.

1.1 Notation

- N: number of teams
- $T = \{t \mid t \in [1, N]\}$: team identifiers
- $P = \{p \mid p \in [0, ..., \frac{N}{2} - 1]\}$: period identifiers.
- $M = \{m \mid m \in [0, ..., \frac{N}{2} - 1]\}$: weekly matchup identifiers.
- $W = \{w \mid w \in [0, ..., N - 2]\}$: week identifiers.
- $S = \{s \mid s \in [0, 1]\}$: slot identifiers, $s = 0$ corresponds to playing at home and $s = 1$ to playing away.
- rb: the precomputed round-robin matrix, where $rb_{m,w,s} = t$ if team $t \in T$ plays in week $w \in W$ in match $m \in M$ and in slot $s \in S$.

2 CP Model

2.1 Decision Variables

This model utilizes two primary decision variable matrices to construct the tournament schedule and assign home/away teams for each match.

2.1.1 matches

For each period $p \in P$ and week $w \in W$

$$matches_{p,w} = m \in M$$

determines that $rb_{m,w}$ is scheduled in week w and period p.

2.1.2 flip_slot

For each period $p \in P$ and week $w \in W$

$$flip\_slot_{p,w} = a \in \{0, 1\}$$

determines which team is assigned as the home team for the match (t_1, t_2) scheduled in week w and period p. Specifically, $a = 0$ means that t_1 plays at home while t_2 away; on the contrary, $a = 1$ means t_2 plays at home and t_1 away.

2.2 Auxiliary Variables

2.2.1 home_games

For each team $t \in T$

$$home\_games_t = h \in [0, \ldots, N - 1]$$

determines the total number of home games assigned to team t throughout the tournament. Its value is derived from the assignments of the decision variables $matches$ and $flip\_slot$.

2.3 Objective Function

The model's objective is to minimize the max_imbalance $\in [1, \ldots, N-1]$ variable quantifying the maximum absolute disparity between home and away game assignments across all teams. The lower bound of 1 is the best possible balance score, since max_imbalance $= 0$ is not achievable for an odd number of games, $N-1$. The upper bound of $N-1$ represents the theoretical maximum possible deviation, occurring if a team plays all its games either at home or away.

The value of max_imbalance is determined by looking for the minimum integer satisfying the following fairness constraints:

$$\forall t \in T : |2 \times \text{home\_games}_t - (N-1)| \leq \text{max\_imbalance}$$

This formulation is derived from the difference between a team's home games and away games: $|\text{home\_games}_t - \text{away\_games}_t|$. Since $\text{home\_games}_t + \text{away\_games}_t = N-1$ (total games), expressing away_games_t as a function of the other quantities and substituting it in the starting expression, yields $2 \times \text{home\_games}_t - (N-1)$. By enforcing that this score must be less than or equal to max_imbalance for every team, we constrain the variable to being grater or equal than the greatest individual team imbalance.

The matchups inside rb have an unbalanced structure: the team with the largest identifier always plays at home, causing all the other teams to also have an imbalanced schedule. To guide the search towards more balanced solutions, we introduce the following constraint:

$$\forall w \in W : \left| \sum_{p \in P} \text{flip\_slot}_{p,w} - \left\lfloor \frac{|P|}{2} \right\rfloor \right| \leq 1.$$

This condition balances, within each week, the flip_slot values, guiding the solver towards assigning home games more equitably, ultimately speeding up the search for an optimal solution.

2.4 Constraints

2.4.1 Core Constraints

These constraints are strictly necessary for defining a feasible round-robin schedule:

1. **Each match is assigned to a unique period each week:**

   $$\forall w \in W : \texttt{all\_different}([\texttt{matches}[p, w] \mid p \in P])$$

   Ensures that for every week, every match of rb is scheduled to no more than one period and every period is assigned to a single match.

2. **Each team plays at most twice in the same period:**

   $$\forall p \in P, \forall t \in T : \left|\{(w, s) \mid w \in W, s \in S, \text{rb}_{\text{matches}_{p,w},w,s} = t\}\right| \leq 2$$

   Implemented using the `global_cardinality` global constraint.

2.4.2 Channeling Constraints

1. **Counting home_games for each team:**

$$\forall t \in T : \text{home\_games}_t = \sum_{p \in P, w \in W, s \in S} \mathbb{I}\left(\text{rb}_{\text{matches}_{p,w},w,s} = t \wedge \text{flip\_slot}_{p,w} = s\right)$$

The indicator function $\mathbb{I}(\cdot)$ ensures that 1 is added to the sum if team t is located in slot s and that slot s is designated as the home slot by $\text{flip\_slot}_{p,w}$.

2.4.3 Implied Constraints

1. **Each team appears exactly once per week:** While this is implicitly ensured by the combination of the `all_different` constraint on `matches` and the structure of `rb`, this constraint helps the solver propagate information earlier and prune the search space more effectively. It is implemented using the `global_cardinality` global constraint:

$$\forall w \in W, \forall t \in T : \left|\{(p,s) \mid p \in P, s \in S, \text{rb}_{\text{matches}_{p,w},w,s} = t\}\right| = 1.$$

2.4.4 Symmetry Breaking Constraints

1. **Period symmetry:** Swapping period labels across weeks yields equivalent schedules, i.e rescheduling games already scheduled for period i to period j and those for period j to i results in another valid assignement. To eliminate this redundancy, we enforce a lexicographic ordering:

$$(\text{matches}_{p,w})_{p \in P, w \in W} \succ_{\text{lex}} (\text{matches}_{p,w})_{\text{reversed}(p) \in P, w \in W}$$

2. **Fix first match home:** Swapping home/away assignments of all matches yields an equivalent optimal solution. To counteract this symmetry, we enforce the following assignment:

$$\text{flip\_slot}_{0,0} = 0$$

2.5 Validation

2.5.1 Experimental Design

The model was implemented in MiniZinc and validated through a series of experiments designed to assess solver performance under various model configurations and search strategies.

Solvers: The following solvers were employed: *Gecode 6.3.0*, *Chuffed 0.13.2* and *OR-Tools CP-SAT 9.12.4544*. The same time limit of 300 seconds was imposed for each individual problem instance.

Model Configurations: Four configurations were tested: `baseline` (core constraints only), `baseline+implied`, `baseline+symmetry breaking`, `full model`.

Search Strategies: Three distinct search strategies were employed to analyze solver behavior

1. **Default Search Strategy (Solver's Default):** Each solver relied entirely on its built-in decision heuristics and restart policies, serving as a baseline for their inherent capabilities.

2. **Sequential Custom Search Strategy:** A manually defined sequential search `seq_search` with `restart_luby(100)` policy was applied. Firstly the `matches` variables are assigned and secondly the `flip_slot` variables. We chose as variable assignment strategies `dom_w_deg` for `matches` and `first_fail` for `flip_slot`, while for value assignments we used `indomain_min` for both.

3. **Relax-and-Reconstruct (LNS) Strategy:** This higher-level strategy incorporates `relax_and_reconstruct` on the `matches` variables (preserving 60% of solution values), leveraging the Large Neighborhood Search (LNS) technique. It was layered on top of the "Sequential Custom Search Strategy".

Solver-Specific Strategy Application: To ensure a fair and controlled comparison under single-threaded conditions (aligning with project constraints), OR-Tools CP-SAT was run without multi-threading. For both Chuffed and OR-Tools CP-SAT, the `free_search` parameter was explicitly omitted when applying the Sequential Custom Search and Relax-and-Reconstruct strategies. This allowed direct evaluation of the user-defined MiniZinc search annotations, rather than the solvers' embedded heuristics.

2.5.2 Experimental Results

In the following tables, we report the objective values found by each solver for the following four model configurations:

- `bs` denotes the baseline model
- `noIMPL` denotes the baseline with symmetry breaking constraints
- `noSB` denotes the baseline with implied constraints
- `complete` denotes the full model with both implied and symmetry breaking constraints

The experimental results, Tables 1 2 3, show that the `CP-SAT` solver, with its default search strategy, consistently delivers the best performance. Moreover, combining Sequential Search with the Relax-and-Reconstruct (LNS) strategy improves `Gecode` by solving larger instances and finding valid or optimal solutions where the default did not succeed.

n	GECODE				CHUFFED				CP-SAT			
	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB
6	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	9	3	3	5	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
12	11	3	3	3	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
14	13	5	5	5	N/A	N/A	4	N/A	**1**	**1**	**1**	**1**
16	15	N/A	N/A	5	N/A	N/A	N/A	N/A	**1**	**1**	**1**	**1**
18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**1**	**1**	**1**	**1**
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**1**	**1**	**1**	**1**
22	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**1**	**1**	**1**

Table 1: Objective values using *Default Search Strategy (Solver's Default)*

n	GECODE				CHUFFED				CP-SAT			
	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB
6	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
12	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
14	**1**	**1**	**1**	**1**	N/A	**1**	**1**	N/A	N/A	N/A	N/A	N/A
16	15	3	**1**	**1**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
22	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 2: Objective values using *Sequential Custom Search Strategy*

n	GECODE				CHUFFED				CP-SAT			
	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB
6	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
10	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
12	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
14	**1**	**1**	**1**	**1**	N/A	N/A	**1**	N/A	N/A	N/A	N/A	N/A
16	**1**	**1**	**1**	**1**	N/A	N/A	13	N/A	N/A	N/A	N/A	N/A
18	N/A	**1**	N/A	**1**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
22	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 3: Objective values using *Relax-and-Reconstruct (LNS) Strategy*

3 SAT Model

Let $X = \{x_1, x_2, ..., x_n\}$ be a set of boolean variables. The constraints are expressed by using the following encodings:

- $AtLeastK(X), AtMostK(X)$ with a sequential encoding
- $ExactlyOne(X) = AtMostOne(X) \wedge AtLeastOne(X)$, where
  - $AtLeastOne(X) = \bigvee_{i=1}^{n} x_i$
  - $AtMostOne(X)$ with Heule Encoding

3.1 Decision variables

Let sts be a schedule analogous to rb satisfying to the STS problem: The satisfiability task is formalized by the following categories of propositions:

3.1.1 Satisfiability

- **match_schedule**
  For each period $p \in P$ and week $w \in W$:

  $$match\_schedule_{p,w,m} = true$$

  if, and only if, the match $rb_{m,w}$ takes place in period p of week w
- **match_to_periods**
  For each period $p \in P$, $t_1 \in T$ and $t_2 \in T/\{t_1\}$:

  $$match\_to\_periods_{t_1,t_2,p} = true$$

  if, and only if, t_1 plays against t_2 in period p.

3.1.2 Optimization

- **flip_slot**
  For each period $p \in P$ and week $w \in W$:

  $$flip\_slot_{p,w} = true$$

  if, and only if, team $sts_{p,w,1} = t_1$ plays away
- **match_to_slots**
  For each period $p \in P$, $t_1 \in T$ and $t_2 \in T/\{t_1\}$:

  $$match\_to\_slots_{t_1,t_2} = true$$

  if, and only if, t_1 plays away and t_2 plays at home.

3.2 Objective function

Similarly to the other approaches, the objective is to minimize the absolute difference between the number of games played at home and away for each team. Let A_t be the number of times the team $t \in T$ plays away, the objective function translates into the following:

$$k^* = \underset{k \in \mathbb{N}}{\operatorname{argmax}} \left(\forall t \in T, A_t \geq k\right)$$

The chosen model allows to solve the optimization task using the *sts* schedule computed in the satisfiability process. The optimization consists of a binary search of k^* in the interval $[1, \lfloor \frac{N-1}{2} \rfloor]$. Since SAT doesn't directly support optimization, a new set of optimizing constraints is introduced for every instance of k. The constraints are defined by first binding $flip\_slots_{p,w}$ to $match\_to\_slots_{p,w}$:

$$\forall p \in P, w \in W.(flip\_slot_{p,w} \leftrightarrow match\_to\_slots_{sts_{p,w,0}, sts_{p,w,1}})$$

and then enforcing, for any k, that t_1 plays at least k times away:

$$\forall p \in P, t_1 \in T.(AtLeastK(match\_to\_slots_{t_1,t_2} | t_2 \in T / \{t_1\}))$$

3.3 Constraints

3.3.1 Each match is assigned to a unique period each week

For each week w we ensure that every match is assigned to one period p

$$\forall p \in P, w \in W.(ExactlyOne(match\_schedule_{p,w,m} | m \in M))$$

and every period is assigned to one match

$$\forall m \in M, w \in W.(ExactlyOne(match\_schedule_{p,w,m} | p \in P))$$

3.3.2 Every team plays at most twice in the same period

First we bind $match\_schedule_{p,w,m}$ to $match\_to\_periods_{t_1,t_2,p}$:

$$\forall p \in P, w \in W, m \in P.(match\_schedule_{p,w,m} \leftrightarrow match\_to\_periods_{rb_{m,w,0}, rb_{m,w,1}, p})$$

and then enforcing that t_1 plays at most 2 times in period p:

$$\forall p \in P, t_1 \in T.AtMost2(match\_to\_periods_{t_1,t_2,p} | t_2 \in T / \{t_1\})$$

3.4 Validation

3.4.1 Experimental design

The model was written in Python by making use of the Z3 4.15.1.0 and the CVC5 1.3.0 libraries, which offers CaDiCaL and MiniSat as the underlying SAT solvers. The objective values found within the 300 second time limit, are presented in Table 4.

3.4.2 Experimental results

All solvers either find the optimal solution or no solution at all. Overall Z3 was the fastest and was able to solve the problems with higher N, reaching $N = 20$. On the other hand CaDiCaL was the worst, failing at $N = 12$, while MiniSat stopped at $N = 14$

n	Z3	MiniSat	CaDiCaL
6	**2**	**2**	**2**
8	**3**	**3**	**3**
10	**4**	**4**	**4**
12	**5**	**5**	N/A
14	**6**	N/A	N/A
16	**7**	N/A	N/A
18	**8**	N/A	N/A
20	**9**	N/A	N/A
22	N/A	N/A	N/A

Table 4: SAT optimization solver results

4 SMT Model

Similarly to the SAT model, the SMT model is structured based on its two tasks: **Satisfiability** and **Optimization**. We encode both the satisfiability and optimization phases in QF_LIA (Quantifier-Free Linear Integer Arithmetic) because it lets us combine Boolean decisions with integer sums and comparisons directly, avoiding complex encodings of counters in pure SAT.

4.1 Decision Variables

4.1.1 Satisfiability

The decision variables for this task are:

- $match\_schedule_{p,w,m} \in \{\text{true}, \text{false}\}$: true if, and only if, the match $m \in M$ is scheduled for period $p \in P$ of week $w \in W$.
- $home_{p,w} = t \in T$: if t is the home team in the match in period $p \in P$ of week $w \in W$.
- $away_{p,w} = t \in T$: if t is the away team in the match in period $p \in P$ of week $w \in W$.

4.1.2 Optimization

To optimize the balance of the number of games played at home and away for each team, the following variables are utilized:

- $flip\_slot_{p,w} \in \{\text{true}, \text{false}\}$: if the value is false, $rb_{p,w,0} = t_1$ plays at home; otherwise it is true, then $rb_{p,w,1} = t_2$ is the one playing at home.
- $home\_eff_{p,w}$, $away\_eff_{p,w} \in T$: are, respectively, the effective home and effective away teams after the slot flip decisions.

4.2 Objective Function

The optimization goal is to minimize the maximum imbalance k between the number of home and away games played by each team, respectively H_t and A_t, throughout the tournament. Formally,

$$k^* = \underset{k \in \mathbb{N}}{\operatorname{argmin}} \left(\forall t \in T. |H_t - A_t| \leq k\right)$$

where:

$$H_t = \sum_{p,w} \mathbb{1}[\, home\_eff_{p,w} = t \,], \quad A_t = \sum_{p,w} \mathbb{1}[\, away\_eff_{p,w} = t \,].$$

The effective home and away variables are computed by considering slot flips:

$$\begin{aligned} home\_eff_{p,w} &= \text{ite}(flip\_slot_{p,w},\, away_{p,w},\, home_{p,w}), \\ away\_eff_{p,w} &= \text{ite}(flip\_slot_{p,w},\, home_{p,w},\, away_{p,w}). \end{aligned}$$

where ite stands for "if then else" statement.

4.3 Constraints

4.3.1 Each match is assigned to a unique period each week

Each period in each week is assigned to exactly one match:

$$\forall p \in P,\, w \in W : \sum_{m \in M} match\_schedule_{p,w,m} = 1.$$

Each match is assigned to exactly one period each week:

$$\forall m \in M,\, w \in W : \sum_{p \in P} match\_schedule_{p,w,m} = 1.$$

4.3.2 Binding team assignments

The teams playing in $match\_schedule_{p,w,m}$ are linked to the corresponding variables $home_{p,w}$ and $away_{p,w}$ based on the values of $rb_{m,w}$ as follows:

$$\forall p \in P,\, w \in W. \big(match\_schedule_{p,w,m} \leftrightarrow (home_{p,w} = rb_{m,w,0} \wedge away_{p,w} = rb_{m,w,1})\big)$$

4.3.3 Every team plays at most twice in the same period

For each team t and each period p, we count how many times t appears either as home or away in that period over all weeks and constraint the count to not exceed 2:

$$\forall t \in T,\, p \in P : \sum_{w \in W} \sum_{m \in M} \left([\, rb_{m,w,0} = t \vee rb_{m,w,1} = t \,] \cdot [\, match\_schedule_{p,w,m} \,] \right) \leq 2.$$

4.3.4 Symmetry breaking

To reduce equivalent permutations of the solutions, the first match is fixed:

$$match\_schedule_{0,0,0} = \text{true}.$$

4.4 Validation

4.4.1 Experimental Design

Solver performance is measured in two stages:

1. **Satisfiability**: Produce a single `.smt2` file that encodes match assignments, period-limit, and symmetry breaking constraints as linear arithmetic formulas. The solver then finds an initial feasible schedule.

2. **Optimization**: Perform a binary search over

   $$k \in \{1, 2, \ldots, N/2\}.$$

   For each candidate k, generate a new `.smt2` file that

   - has the base *home*/*away* assignments,
   - adds boolean `flip_slot` variables,
   - defines effective $home\_eff, away\_eff$ via `ite`,
   - asserts $|H_t - A_t| \leq k$ using integer counters H_t, A_t.

   This produces $\log_2(N) - 1$ optimization files.

The total runtime is measured from the start of the satisfiability phase until either a solution for $k = 1$ is found or the 300 s timeout is reached. We tested the following solvers: Z3 4.15.1.0 and CVC5 1.3.0 and present our findings in table 5.

4.4.2 Experimental results

The experimental results show that Z3 performs significantly better than CVC5 in terms of solving time and scalability. Specifically, Z3 solved instances up to $N = 22$, while CVC5 handled only smaller instances up to $N = 8$.

N	Z3	CVC5
6	**1**	**1**
8	**1**	**1**
10	**1**	N/A
12	**1**	N/A
14	**1**	N/A
16	**1**	N/A
18	**1**	N/A
20	**1**	N/A
22	**1**	N/A
24	N/A	N/A

Table 5: SMT optimization solver results

5 MIP

5.1 Decision variables

5.1.1 match_schedule

For each $w \in W$ and $p \in P$, the binary

$$match\_schedule_{w,p,m} = 1$$

if, and only if, in week w and period p match $m \in M$ is played

5.1.2 flip_slot

For each week $w \in W$ and period $p \in P$

$$flip\_slot_{w,p} = a \in \{0, 1\}$$

determines which team from match $rb_{m,w} = (t_1, t_2)$ in week w and period p is assigned as the home team. Specifically, $a = 0$ means t_1 plays at home and t_2 away, while $a = 1$ means the assignment is reversed.

5.1.3 team periods of play

For each $t \in T$, $w \in W$ and $p \in P$, the binary

$$TP_{t,w,p} = 1$$

if, and only if, team t plays in week w and period p.

5.1.4 home games counter

For each team $t \in T$, the integer

$$H_t = h \in [0, \ldots, N-1]$$

determines the total number of home games assigned to team t throughout the tournament.

5.2 Objective Function

The model's objective is to minimize the maximum absolute difference between the number of home and away games played by any team. This maximum difference is quantified by the integer variable $D \in [1, \ldots, N-1]$, which was previously referred to as max_imbalance. We first find, for each team, the number of games played at home (2), and then constrain D as being greater or equal than the difference of home and away games for each team. Given that we have a minimization problem, this is effectively equivalent to computing the max:

$$\begin{cases} \min D \\ D = \max_t |home_t - away_t| \end{cases} = \begin{cases} \min D \\ D \geq |home_t - away_t| \ \forall t \in T \end{cases}$$

The absolute value of the difference is not computed explicitly but is decomposed into 2 inequalities (3)(4). Finally, we look for the minimum of D (1).

$$\min D \tag{1}$$

$$H_t = \sum_{w \in W, m \in M} \mathbb{1}[flip\_slot_{w,p} = 0 \wedge rb_{m,w,0} = t] + \sum_{w \in W, m \in M} \mathbb{1}[flip\_slot_{w,p} = 1 \wedge rb_{m,w,1} = t] \qquad \forall t \in T \tag{2}$$

$$D \geq 2H_t - (N-1) \qquad \forall t \in T \tag{3}$$

$$D \geq -(2H_t - (N-1)) \qquad \forall t \in T \tag{4}$$

5.3 Constraints

5.3.1 Each match is assigned to a unique period each week

Due to how the decision variables are defined, it was necessary to impose that each period in each week is assigned a single match (5) and each match is assigned to a single period (6).

$$\sum_{m \in M} match\_schedule_{w,p,m} = 1 \qquad \forall p \in P \quad \forall w \in W \tag{5}$$

$$\sum_{p \in P} match\_schedule_{w,p,m} = 1 \qquad \forall m \in M \quad \forall w \in W \tag{6}$$

5.3.2 Each team plays at most twice in the same period

The constraint is imposed by first linking the variables of $match\_schedule_{w,p,m}$ to those in $TP_{t,w,p}$ based on the values in $rb_{m,w}$ (7) and then limiting the sum of periods of play to being smaller or equal than 2 (8).

$$TP_{t,w,p} = match\_schedule_{w,p,m} \qquad \text{where } t \in rb_{m,w} \qquad \forall t \forall p \forall w \tag{7}$$

$$\sum_{w \in W} TP_{t,w,p} \leq 2 \qquad \forall t \in T \quad \forall p \in P \tag{8}$$

5.4 Validation

Experimental design

The model is written in Python by making use of the PuLP library. The solvers tested on the model were: CBC 2.10.3, HiGHS 1.10.0, CPLEX 22.1.1 and SCIP 5.5.0, all used with their default parameters. The time elapsed to find an optimal solution, within the 300 second time limit, was measured for each solver up to $N = 22$.

Experimental results

As shown in Table 6, CBC has the worst performance, it isn't able to find the optimal solution for N greater than 14. CPLEX, instead, stops at N = 16 while HiGHS at N=18. SCIP is the best performing model, finding optimal solutions up to N=20 and stopping after that.

N	**CBC**	**HiGHS**	**CPLEX**	**SCIP**
6	**1**	**1**	**1**	**1**
8	**1**	**1**	**1**	**1**
10	**1**	**1**	**1**	**1**
12	**1**	**1**	**1**	**1**
14	**1**	**1**	**1**	**1**
16	N/A	**1**	**1**	**1**
18	N/A	**1**	N/A	**1**
20	N/A	N/A	N/A	**1**
22	N/A	N/A	N/A	N/A

Table 6: MIP optimization solver results

6 Conclusions

This report explored the Sports Tournament Scheduling (STS) problem with CP, SAT, SMT and MIP. Across all approaches, the precomputation of the

round-robin tournament matrix significantly sped up the search process by reducing the search space and ensuring fundamental schedule properties.

While CP, SAT, SMT, and MIP models generally yielded comparable results, CP and SMT demonstrated slightly better performance. OR-Tools CP-SAT, for the CP model, and Z3, for the SMT model, were able to solve instances up to N=22 to optimality within the time limits. In contrast, the best performing SAT solver, Z3, reached N=20, and the best MIP solver, SCIP, also reached N=20.

The experimental results presented in this report were conducted under single-threaded conditions. Therefore, a promising avenue for further performance improvement would be to explore the benefits of parallelism, leveraging multi-threading capabilities to potentially solve larger instances or achieve faster optimal solutions.

Authenticity and Author Contribution Statement We declare that the work presented in this report is our own and has not been copied from any external source, except where explicitly cited. All modelling decisions, encodings, implementations and experiments were carried out by us without unauthorized assistance. All the modelling decisions were agreed on as a group, while the implementations were carried out more autonomously:

- Cono Cirone: CP
- Giulio Petrozziello: SAT
- Jacopo Dardini: SMT
- Gio Formichella: MIP

References

[1] P. Van Hentenryck, L. Michel, L. Perron, and J. C. Régin. Constraint programming in opl. In Gopalan Nadathur, editor, *Principles and Practice of Declarative Programming*, pages 98–116, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[2] J. H. Dinitz, D. Froncek, E. R. Lamken, and W. D. Wallis. Scheduling a tournament. In Charles J. Colbourn and Jeffrey H. Dinitz, editors, *Handbook of Combinatorial Designs*, pages 591–598, Boca Raton, FL, 2006. CRC Press.

[3] E. R. Lamken. Balanced tournament designs. In Charles J. Colbourn and Jeffrey H. Dinitz, editors, *Handbook of Combinatorial Designs*, pages 333–336, Boca Raton, FL, 2006. CRC Press.