Sports Tournament Scheduling problem

Hassan Mujtaba Ahmed
hassan.ahmed5@studio.unibo.it

Muhammad Talha Sohail Chattha
muhammadtalha.chatha@studio.unibo.it

November 2025

Contents

1 Introduction 3

2 Constraint Programming (CP) Model 4
- 2.1 Decision variables . . . 4
- 2.2 Hard constraints . . . 4
  - 2.2.1 No self-play . . . 4
  - 2.2.2 Each team appears exactly once per week . . . 4
  - 2.2.3 Pairwise uniqueness . . . 5
  - 2.2.4 Period fairness . . . 5
  - 2.2.5 Home/away balance . . . 5
  - 2.2.6 Symmetry breaking . . . 5
- 2.3 Search and implementation notes . . . 5
- 2.4 CP results (summary) . . . 5

3 SAT Model 7
- 3.1 Model Overview . . . 7
- 3.2 Sequential Counter Encodings . . . 7
- 3.3 Constraints . . . 7
  - 3.3.1 No Self-Play . . . 7
  - 3.3.2 Weekly Uniqueness . . . 8
  - 3.3.3 Unique Pair Matches . . . 8
  - 3.3.4 Period Limit . . . 8
  - 3.3.5 Home/Away Balance . . . 8
  - 3.3.6 Symmetry Breaking . . . 8
- 3.4 Implementation and Solver . . . 8
- 3.5 Results . . . 8

4 Mixed Integer Programming (MIP) Model 9
- 4.1 Model overview . . . 9
- 4.2 Decision variables . . . 9
- 4.3 Constraints . . . 9
  - 4.3.1 Slot assignment . . . 9
  - 4.3.2 No self-match . . . 9
  - 4.3.3 Each team plays exactly once per week . . . 10
  - 4.3.4 Linearised pairwise uniqueness . . . 10
  - 4.3.5 Period limit . . . 10
  - 4.3.6 Home/away balance . . . 10
  - 4.3.7 Symmetry breaking . . . 10
- 4.4 Complexity and model size . . . 10
- 4.5 MIP results . . . 10

4.6 Discussion of MIP behaviour 11

5 Experimental Comparison and Discussion 12

5.1 Aggregate comparison 12

5.2 Observations 12

6 Conclusions 13

Author 14

References 15

Chapter 1

Introduction

The Sports tournament scheduling problem requires arranging n even teams so that each team plays every other team exactly once over $W = n - 1$ weeks. Each week contains $P = n/2$ periods. The challenge is to compute a feasible schedule satisfying structural fairness and operational constraints.

Practical constraints imposed include:

- every team plays exactly once per week;
- no team plays itself;
- every unordered pair of teams meets exactly once in the tournament;
- a team appears in a given period at most twice during the tournament;
- home/away games for each team are balanced.

The Sports tournament scheduling problem is combinatorial and known to be computationally challenging as n grows. Different modelling paradigms exploit different reasoning techniques:

Constraint Programming (CP) relies on finite-domain integer variables, global constraints (e.g. `alldifferent`), and propagation with search heuristics.

SAT encoding translates assignments into Boolean literals and enforces cardinalities using auxiliary Boolean variables to obtain compact, propagating encodings.

Mixed Integer Programming (MIP) models the problem with binary variables and linear constraints, often requiring auxiliary variables to linearise non-linear constructs (e.g., pairwise match detection).

This report provides: (i) formal descriptions of the three models, (ii) implementation notes and solver settings, and (iii) experimental results comparing runtimes and solver behaviour for $n = 6, 8, 10$.

Chapter 2

Constraint Programming (CP) Model

2.1 Decision variables

Let the set of teams be $T = \{1, \dots, n\}$, with n even Weeks. Weeks and periods are:

$$W = \{1, \dots, n-1\}, \qquad P = \{1, \dots, n/2\}.$$

We use the integer decision variable

$$\text{schedule}[p, w, s] \in T, \qquad p \in P,\ w \in W,\ s \in \{1, 2\}, \tag{2.1}$$

where slot $s = 1$ denotes the *home* team and $s = 2$ denotes the *away* team.

For compactness we also use indicator expressions such as $[\text{schedule}[p, w, s] = t]$ to denote a Boolean (1 if true, 0 otherwise) used inside sums for counting constraints.

2.2 Hard constraints

All constraints below are enforced as hard constraints.

2.2.1 No self-play

$$\forall p \in P,\ \forall w \in W : \qquad \text{schedule}[p, w, 1] \neq \text{schedule}[p, w, 2]. \tag{2.2}$$

2.2.2 Each team appears exactly once per week

Each team must appear exactly once in each week. Using an `alldifferent` global constraint:

$$\forall w \in W : \qquad \text{alldifferent}\Big(\{\text{schedule}[p, w, s] \mid p \in P,\ s \in \{1, 2\}\}\Big). \tag{2.3}$$

Equivalently this enforces that for every team t and every week w:

$$\sum_{p \in P} \sum_{s=1}^{2} [\text{schedule}[p, w, s] = t] = 1. \tag{2.4}$$

2.2.3 Pairwise uniqueness

Every unordered pair $\{i, j\}$ must meet exactly once across all weeks:

$$\begin{aligned}\forall i < j : \quad \sum_{p \in P} \sum_{w \in W} \Big(& [\text{schedule}[p, w, 1] = i]\, [\text{schedule}[p, w, 2] = j] + \\ & [\text{schedule}[p, w, 1] = j]\, [\text{schedule}[p, w, 2] = i] \Big) = 1.\end{aligned} \tag{2.5}$$

2.2.4 Period fairness

A team should not appear more than twice in the same period across all weeks:

$$\forall t \in T,\ \forall p \in P : \qquad \sum_{w \in W} \sum_{s=1}^{2} [\text{schedule}[p, w, s] = t] \le 2. \tag{2.6}$$

2.2.5 Home/away balance

Let H_t and A_t denote the number of home and away games for team t across all weeks:

$$H_t = \sum_{p \in P} \sum_{w \in W} [\text{schedule}[p, w, 1] = t], \qquad A_t = \sum_{p \in P} \sum_{w \in W} [\text{schedule}[p, w, 2] = t].$$

We enforce balance within one:

$$\forall t \in T : \qquad |H_t - A_t| \le 1. \tag{2.7}$$

2.2.6 Symmetry breaking

To reduce symmetric solutions we fix an ordering on week 1:

$$\forall p \in P : \qquad \text{schedule}[p, 1, 1] < \text{schedule}[p, 1, 2]. \tag{2.8}$$

2.3 Search and implementation notes

Our MiniZinc implementation uses array variables and the ‘alldifferent‘ constraint. We tested three different search strategies:

1. **default:** `solve satisfy;`
2. **first fail:** `int_search(schedule, first_fail, indomain_min);`
3. **dom/wdeg:** `int_search(schedule, dom_wdeg, indomain_random);`

Solvers: Gecode and Chuffed (MiniZinc solvers).

2.4 CP results (summary)

Table 2.1 reports the observed runtimes for the CP experiments (Gecode and Chuffed) on instances $n = \{6, 8, 10\}$.

Teams n	**Strategy**	**Time (s)**	**Optimal**	**Solution Found**
6	Default	0	Yes	Yes
6	First Fail	0	Yes	Yes
6	Dom w/ Deg	0	Yes	Yes
8	Default	0	Yes	Yes
8	First Fail	27	Yes	Yes
8	Dom w/ Deg	0	Yes	Yes
10	Default	300	No	-
10	First Fail	5	Yes	Yes
10	Dom w/ Deg	61	Yes	Yes
Chuffed (default)				
6		0	Yes	Yes
8		1	Yes	Yes
10		300	No	-

Table 2.1: CP solver results for different strategies and number of teams.Time limit: 300 s

Chapter 3

SAT Model

3.1 Model Overview

The SAT encoding uses integer schedule variables, but constraints are expressed as Boolean combinations, emulating a pure SAT approach. Each variable schedule$[p, w, s]$ takes integer values from 1 to n, but all constraints are reformulated using Boolean literals:

$$\ell_{p,w,s,t} \equiv (\text{schedule}[p, w, s] = t), \quad t \in 1 \dots n.$$

Constraints are implemented via sequential counters, a method for efficiently encoding cardinality restrictions with auxiliary Boolean variables.

3.2 Sequential Counter Encodings

Definitions

For a set of Boolean variables $\{x_1, \dots, x_m\}$, sequential counters encode:

- **at_most_one**: $\sum_i x_i \leq 1$
- **at_least_one**: $\sum_i x_i \geq 1$
- **exactly_one**: $\sum_i x_i = 1$
- **at_most_k**: $\sum_i x_i \leq k$
- **at_least_k**: $\sum_i x_i \geq k$
- **exactly_k**: $\sum_i x_i = k$

Auxiliary variables $s_{i,j}$ track the cumulative count of True variables up to position i, enabling propagation and efficient SAT reasoning.

3.3 Constraints

3.3.1 No Self-Play

$$\ell_{p,w,0,t} \Rightarrow \neg \ell_{p,w,1,t}, \quad \forall t$$

3.3.2 Weekly Uniqueness

Each team plays exactly once per week:

$$\text{exactly\_one\_seq}(\{\ell_{p,w,s,t} \mid p = 1 \dots P, s = 0, 1\})$$

3.3.3 Unique Pair Matches

Each pair of teams $\{t_1, t_2\}$ meets exactly once:

$$\text{exactly\_one\_seq}(\{\ell_{p,w,0,t_1} \wedge \ell_{p,w,1,t_2} \ \vee \ \ell_{p,w,0,t_2} \wedge \ell_{p,w,1,t_1}\})$$

3.3.4 Period Limit

No team appears more than twice per period:

$$\text{at\_most\_k\_seq}(\{\ell_{p,w,s,t} \mid w, s\}, 2)$$

3.3.5 Home/Away Balance

Home and away game counts are constrained:

$$\text{LowerBound} \leq \sum_{p,w} \ell_{p,w,0,t} \leq \text{UpperBound}, \quad \text{LowerBound} \leq \sum_{p,w} \ell_{p,w,1,t} \leq \text{UpperBound}$$

3.3.6 Symmetry Breaking

First week order fixed:

$$\ell_{p,0,0,t} \Rightarrow \neg \ell_{p,0,1,t}, \quad \forall p$$

3.4 Implementation and Solver

The model was implemented in Python using Z3, translating integer variables to Boolean literals for sequential counters. Each constraint was encoded as described above. The solver enforced all constraints and generated feasible schedules.

3.5 Results

Teams n	**Time (s)**	**Optimal**	**Solution Found**	**Notes**
6	0	Yes	Yes	Instant
8	0	Yes	Yes	Instant
10	7	Yes	Yes	Slightly longer due to larger search space

Table 3.1: Runtime of the SAT model using Z3.

Chapter 4

Mixed Integer Programming (MIP) Model

4.1 Model overview

The MIP model uses binary variables to represent team assignments into slots and linear constraints to represent all logical relations. Non-linear interactions (product of binaries) are linearised using auxiliary binary variables and standard big-M constraints. The formulation was implemented in Python using PULP and solved with the open-source CBC solver.

4.2 Decision variables

Main binary variables:

$$s_{p,w,s,t} \in \{0,1\}, \tag{4.1}$$

with $s_{p,w,s,t} = 1$ iff team t occupies slot s in period p during week w.

Auxiliary match-detection binaries are introduced to linearise conjunctions:

$$m1_{p,w,t_1,t_2}, \quad m2_{p,w,t_1,t_2} \in \{0,1\}, \tag{4.2}$$

representing whether (t_1, t_2) meet in the oriented combinations (home/away) at (p, w).

4.3 Constraints

4.3.1 Slot assignment

Each slot must be occupied by exactly one team:

$$\forall p, w, s : \qquad \sum_t s_{p,w,s,t} = 1. \tag{4.3}$$

4.3.2 No self-match

$$\forall p, w, t : \qquad s_{p,w,0,t} + s_{p,w,1,t} \leq 1. \tag{4.4}$$

4.3.3 Each team plays exactly once per week

$$\forall w, t: \qquad \sum_p \sum_s s_{p,w,s,t} = 1. \tag{4.5}$$

4.3.4 Linearised pairwise uniqueness

Introduce $m1_{p,w,t_1,t_2}$ with constraints:

$$m1_{p,w,t_1,t_2} \le s_{p,w,0,t_1}, \tag{4.6}$$
$$m1_{p,w,t_1,t_2} \le s_{p,w,1,t_2}, \tag{4.7}$$
$$m1_{p,w,t_1,t_2} \ge s_{p,w,0,t_1} + s_{p,w,1,t_2} - 1, \tag{4.8}$$

(and similarly for $m2$ with swapped indices). Then enforce:

$$\forall t_1 < t_2: \qquad \sum_{p,w} (m1_{p,w,t_1,t_2} + m2_{p,w,t_1,t_2}) = 1. \tag{4.9}$$

4.3.5 Period limit

$$\forall t, p: \qquad \sum_w \sum_s s_{p,w,s,t} \le 2. \tag{4.10}$$

4.3.6 Home/away balance

Let $H_t = \sum_{p,w} s_{p,w,0,t}$ and $A_t = \sum_{p,w} s_{p,w,1,t}$. Enforce:

$$-H_t + A_t \le 1, \qquad H_t - A_t \le 1. \tag{4.11}$$

4.3.7 Symmetry breaking

We linearise the ordering constraint on week 1 by introducing team indices multiplied by binaries:

$$\sum_t t \cdot s_{p,1,0,t} \le \sum_t t \cdot s_{p,1,1,t} - 1, \qquad \forall p. \tag{4.12}$$

This is linear because t is a constant and $s_{p,1,*,t}$ are binaries.

4.4 Complexity and model size

The binary variable count is $n^2(n-1)$ for $s_{p,w,s,t}$ plus $O(n^2PW)$ auxiliaries for m variables. The number of constraints grows as $O(n^4)$ in the worst-case due to all pairwise linearisation terms, which explains the sharp runtime increase as n grows.

4.5 MIP results

Table 4.1 reports the runtimes obtained with PULP + CBC (300 s timeout).

n	Time (s)	Optimal	Solution Found
6	0	Yes	Yes
8	118	Yes	Yes
10	300	No	-

Table 4.1: MIP (PULP + CBC) results. Time limit: 300 s.

4.6 Discussion of MIP behaviour

The linearisation of conjunctions (pairwise matches) creates a large number of auxiliary variables and weak LP relaxations. While MIP provides a mathematically clean linear model, its LP bound strength for this combinatorial structure is insufficient to avoid extensive branching. Commercial solvers such as Gurobi often perform much better; however, with the open-source CBC solver the $n = 10$ instance times out.

Chapter 5

Experimental Comparison and Discussion

5.1 Aggregate comparison

Tables 2.1, 3.1 and 4.1 summarize runtimes. For clarity we present a condensed comparison table (Table 5.1).

n	CP (best)	SAT (Z3)	MIP (CBC)
6	0	0	0
8	0	0	118
10	5	7	300 (timeout)

Table 5.1: Best observed runtimes across seconds. For CP the best across tested strategies is reported.

5.2 Observations

- **Small instances ($n \leq 8$):** All paradigms can find feasible schedules quickly. SAT and CP (with good heuristics) often find solutions instantly.
- **Medium instances ($n = 10$):** CP with tuned search heuristics and SAT encoding is still competitive. MIP with CBC suffers from model size and weak LP bounds and hits the timeout.
- **Effect of encoding:** Sequential counters in the SAT encoding provide strong propagation for cardinality constraints and scale linearly in auxiliary variables, making the SAT approach efficient for these cardinality-heavy constraints.
- **Solver-dependent heuristics:** CP performance is sensitive to branching heuristics (first-fail vs dom/wdeg). SAT models rely on the underlying SAT engine's clause learning and propagation; Z3 performed still well here.

Chapter 6

Conclusions

This report compared CP, SAT, and MIP formulations of the Sports tournament scheduling problem. We provided formal models, implementation notes, and results on representative instances.

Key conclusions:

- CP and SAT encodings handled the tested instances effectively; SAT encoding (Z3) solved $n = 10$ in $7\,\mathrm{s}$ while MIP (CBC) timed out.
- Sequential counters give SAT encodings a practical edge when many cardinality constraints must be enforced.
- MIP linearization produces large models with weak relaxations for this problem; commercial MIP solvers may reduce this but are not always available.

Author Contribution

The project was completed at the end of November. Both the participants worked on all the parts of the problem.

Bibliography

[1] Carsten Sinz, “Towards an optimal CNF encoding of boolean cardinality constraints,” *Principles and Practice of Constraint Programming*, 2005.

[2] MiniZinc Team, *MiniZinc: A Free Constraint Modelling Language*, `https://www.minizinc.org`.

[3] L. de Moura and N. Bjørner, “Z3: An Efficient SMT Solver,” *Tools and Algorithms for the Construction and Analysis of Systems*, 2008.