Sports Tournament Scheduling: A Comparative Study using CP, SAT, and MIP

Ali, Saleh Mohammadabad
ali.saleh4@studio.unibo.it

Saleh, Mir Mohammad Rezaei
sa.mirmohammadrezaei@studio.unibo.it

February 1, 2026

Abstract

This report presents a comprehensive study of the Sports Tournament Scheduling (STS) problem using three optimization paradigms: Constraint Programming (CP), propositional Satisfiability (SAT), and Mixed-Integer Programming (MIP). We model the problem of scheduling n teams over $n-1$ weeks with $n/2$ periods per week, ensuring each pair of teams plays exactly once while respecting period usage limits. We implement decision versions, comparing the performance and scalability of different approaches on instances up to $n = 20$ teams. The results reveal significant scalability limitations across all approaches. While small instances can be solved, performance degrades quickly as n increases. For $n = 12$, CP performance differs markedly across solvers: Chuffed finds solutions within seconds, whereas Gecode fails to solve the instance within the time limit; SAT and MIP achieve limited success at this size. None of the approaches is able to solve instances with $n = 14$, highlighting the computational difficulty of larger problem sizes.

1 Introduction

The Sports Tournament Scheduling (STS) problem is a well-studied combinatorial optimization problem with practical applications in organizing sports leagues and competitions. The problem involves creating a schedule that satisfies multiple constraints.

1.1 Problem formalization

Input parameters

- n: number of teams (even integer, $n \geq 2$)

- $W = n - 1$: number of weeks in the tournament
- $P = \frac{n}{2}$: number of periods per week

Problem structure Teams participate in a round-robin tournament where each pair of teams plays exactly once. The tournament spans W weeks, each divided into P time periods. In each period, one game is played with one team designated as the home team and the other as the away team. The total number of games is $\frac{n(n-1)}{2} = W \cdot P$.

Constraints

1. **No self-play**: A team cannot play against itself
2. **One game per week**: Each team plays exactly once per week
3. **Round-robin**: Each pair of teams plays exactly once in the tournament
4. **Period limit**: Each team plays in the same period at most twice across all weeks

Objective

- **Decision version**: Find any valid schedule satisfying all constraints

2 CP Model

2.1 Decision variables

We define two matrices of decision variables:

- $\text{Home}[p, w] \in \{1, \ldots, n\}$: the team playing at home in period p of week w
- $\text{Away}[p, w] \in \{1, \ldots, n\}$: the team playing away in period p of week w

for all $p \in \{1, \ldots, P\}$ and $w \in \{1, \ldots, W\}$. This representation directly models the schedule structure, where each matrix entry corresponds to one slot of a game.

The model contains $2 \cdot P \cdot W = n(n-1)$ integer variables, each with domain $\{1, \ldots, n\}$. For example, with $n = 6$ teams, we have 30 variables with domain size 6.

2.2 Objective function

For the decision version, we seek any feasible solution:

$$\text{solve satisfy}$$

For the optimization version, we minimize the total imbalance in home/away games:

$$\text{minimize} \quad \sum_{t=1}^{n} |\text{HomeCount}[t] - \text{AwayCount}[t]|$$

where:

$$\text{HomeCount}[t] = \sum_{p=1}^{P} \sum_{w=1}^{W} [\text{Home}[p, w] = t]$$
$$\text{AwayCount}[t] = \sum_{p=1}^{P} \sum_{w=1}^{W} [\text{Away}[p, w] = t]$$

Here, $[\cdot]$ denotes the Iverson bracket (1 if condition is true, 0 otherwise). Since each team plays $n - 1$ games total, perfect balance means $\frac{n-1}{2}$ home and $\frac{n-1}{2}$ away games.

2.3 Constraints

2.3.1 Main problem constraints

C1: No self-play

$$\text{Home}[p, w] \neq \text{Away}[p, w] \qquad \forall p \in \{1, \ldots, P\},\ w \in \{1, \ldots, W\}$$

This ensures that in each time slot, the home and away teams are different.

C2: Each team plays once per week

$$\sum_{p=1}^{P} ([\text{Home}[p, w] = t] + [\text{Away}[p, w] = t]) = 1 \qquad \forall t \in \{1, \ldots, n\},\ w \in \{1, \ldots, W\}$$

This ensures each team participates in exactly one game per week (either home or away).

C3: Each pair plays once (round-robin)

$$\sum_{w=1}^{W} \sum_{p=1}^{P} ([\text{Home}[p, w] = i \wedge \text{Away}[p, w] = j] + [\text{Home}[p, w] = j \wedge \text{Away}[p, w] = i]) = 1$$

$$\forall i, j \in \{1, \ldots, n\},\ i < j$$

This enforces the round-robin structure where every pair of teams plays exactly once (in either orientation).

C4: Period usage limit

$$\sum_{w=1}^{W} ([\text{Home}[p,w] = t] + [\text{Away}[p,w] = t]) \leq 2 \qquad \forall t \in \{1,\ldots,n\},\, p \in \{1,\ldots,P\}$$

This prevents any team from being scheduled in the same period more than twice across the entire tournament.

2.3.2 Symmetry breaking constraints

To reduce the search space, we impose two symmetry breaking constraints:

SB1: Fix first week pairings

$$\text{Home}[p,1] = p \quad \text{and} \quad \text{Away}[p,1] = n - p + 1 \qquad \forall p \in \{1,\ldots,P\}$$

This fixes the first week to a canonical pattern: $(1 \text{ vs } n), (2 \text{ vs } n{-}1), \ldots, (P \text{ vs } P{+}1)$, breaking symmetries related to week permutation and initial team pairing.

SB2: Team 1 always plays home against larger teams

$$\neg(\text{Home}[p,w] = t \wedge \text{Away}[p,w] = 1) \qquad \forall t \in \{2,\ldots,n\},\, p \in \{1,\ldots,P\},\, w \in \{1,\ldots,W\}$$

This constraint ensures that when team 1 plays against any team $t > 1$, team 1 must be at home. This breaks home/away symmetry without over-constraining the problem.

2.4 Validation

2.4.1 Experimental design

The CP model is implemented in MiniZinc and evaluated using two solvers: Gecode (version 6.3.0) and Chuffed. Gecode is a state-of-the-art CP solver that provides efficient propagation mechanisms and a variety of built-in search strategies, while Chuffed is a lazy-clause-generation solver designed to combine constraint programming with SAT-style learning. The solver is configured with:

- Time limit: 300 seconds
- Default search strategy (variable and value selection)
- Single-threaded execution

All experiments were conducted on a MacBook Air equipped with an Apple M2 processor and 8 GB of RAM, running macOS. Instances tested: $n \in \{6, 8, 10, 12, 14, 16, 18, 20\}$.

	Gecode		**Chuffed**	
n	**Time (s)**	**Status**	**Time (s)**	**Status**
4	300	UNSAT	300	UNSAT
6	**¡ 1**	Solution found	**¡ 1**	Solution found
8	**¡ 1**	Solution found	**¡ 1**	Solution found
10	300	Timeout	**¡ 1**	Solution found
12	300	Timeout	**¡ 2**	Solution found
14	300	Timeout	300	Timeout
16	300	Timeout	300	Timeout
18	300	Timeout	300	Timeout
20	300	Timeout	300	Timeout

Table 1: CP results with Gecode.

2.4.2 Experimental results

Table 1 presents the results for decision versions using Gecode and Chuffed.

Within the CP approach, solver performance varies significantly. For small instances, both Gecode and Chuffed are able to find solutions quickly; however, as the problem size increases, their behavior diverges. Gecode struggles to scale and fails to solve instances beyond small values of n within the time limit, whereas Chuffed remains effective on moderately larger instances before also reaching scalability limits. Although symmetry-breaking constraints are applied to reduce the search space, they are not sufficient to ensure efficient solving for larger instances across either solver.

3 SAT Model

3.1 Decision variables

We define Boolean literals (propositional variables) $M_{i,j,p,w}$ for all $i, j \in \{0, \ldots, n-1\}$ with $i \neq j$, $p \in \{0, \ldots, P-1\}$, and $w \in \{0, \ldots, W-1\}$, where:

$$M_{i,j,p,w} = \text{true} \iff \text{team } i \text{ plays at home against team } j \text{ in period } p \text{ of week } w$$

The model contains $n(n-1) \cdot P \cdot W = \frac{n^2(n-1)^2}{2}$ Boolean variables. For example, with $n = 6$ teams, we have 450 Boolean literals. This is the same variable structure as the MIP model, but using Boolean logic rather than integer programming.

3.2 Objective function

For the decision version, we seek any assignment of Boolean values that satisfies all constraints. SAT solvers naturally solve decision problems; there is no objective function.

3.3 Constraints

The SAT model encodes all constraints using propositional logic formulas, leveraging Z3's Pseudo-Boolean (PB) constraint support for efficiency.

3.3.1 Main problem constraints

C1: One match per slot For each period p and week w, exactly one ordered match is scheduled in that slot:

$$\mathrm{PbEq}\Big(\{(M_{i,j,p,w},1) \;:\; i,j \in \{0,\dots,n-1\},\; i \neq j\},\; 1\Big),$$

i.e., $\sum_{i \neq j} M_{i,j,p,w} = 1$.

C2: Each team plays once per week For each team t and week w, team t appears in exactly one match in that week (either as first or second component, across all periods):

$$\mathrm{PbEq}\Big(\{(M_{t,j,p,w},1) \;:\; j \neq t,\; p \in \{0,\dots,P-1\}\} \cup \{(M_{i,t,p,w},1) \;:\; i \neq t,\; p \in \{0,\dots,P-1\}\},\; 1\Big).$$

C3: Each pair plays once For each unordered pair of teams (i,j) with $i < j$, the two teams meet exactly once across the whole season, in either orientation:

$$\mathrm{PbEq}\Big(\{(M_{i,j,p,w},1),(M_{j,i,p,w},1) \;:\; p \in \{0,\dots,P-1\},\; w \in \{0,\dots,W-1\}\},\; 1\Big).$$

C4: Period usage limit For each team t and period p, the team can appear in that period in at most 2 weeks (again in either orientation):

$$\mathrm{AtMost}\Big(\{M_{t,j,p,w} \;:\; j \neq t,\; w \in \{0,\dots,W-1\}\} \cup \{M_{i,t,p,w} \;:\; i \neq t,\; w \in \{0,\dots,W-1\}\},\; 2\Big).$$

3.3.2 Implied constraints

Several constraints are implicitly enforced by the combination of the main constraints. In particular, constraint C1 ensures that no slot contains more than one match, while constraints C2 and C3 together prevent a team from playing multiple matches in the same week or playing the same opponent more than once. As a result, no additional explicit constraints are required to forbid self-matches or duplicate assignments.

3.3.3 Symmetry considerations

The SAT model does not include explicit symmetry-breaking constraints. Symmetries related to home/away orientation and permutations of weeks or periods are left to the solver. Instead, the model relies on Z3's internal reasoning and the strength of the Pseudo-Boolean constraints to handle symmetric solutions implicitly.

3.4 Validation

3.4.1 Experimental design

The SAT model is implemented in Python using the Z3 SMT solver (version 4.15.4), relying on Z3's native support for Pseudo-Boolean constraints in a SAT-based solving configuration. The following settings are used:

- Timeout: 300 seconds (300,000 milliseconds)
- Default Z3 search configuration
- Single-threaded execution

All experiments were conducted on a MacBook Air equipped with an Apple M2 processor and 8 GB of RAM, running macOS, using single-threaded execution. The tested instances correspond to problem sizes $n \in \{4, 6, 8, 10, 12, 14, 16, 18, 20\}$.

3.4.2 Experimental results

Table 2 reports the results obtained with Z3 for the decision version of the SAT model.

n	**Time (s)**	**Status**
4	300	UNSAT
6	**¡ 1**	SAT
8	**¡ 1**	SAT
10	**¡ 4**	SAT
12	**¡ 46**	SAT
14	300	Timeout
16	300	Timeout
18	300	Timeout
20	300	Timeout

Table 2: SAT results obtained with Z3.

The SAT approach is able to find feasible solutions for small instances, up to $n = 12$, within the imposed time limit. The instance with $n = 4$ is correctly identified as unsatisfiable. For larger instances, starting from $n = 14$, Z3 fails to find a solution within the allotted 300 seconds, indicating a clear scalability limitation of the SAT encoding.

4 MIP Model

4.1 Decision variables

We define binary decision variables $M_{i,j,p,w} \in \{0, 1\}$ with the same semantics as the SAT model, but formulated for integer programming.

4.2 Constraints

All constraints are formulated as linear inequalities and equations:

C1: One match per slot

$$\sum_{\substack{i,j\in\{0,\ldots,n-1\}\\ i\neq j}} M_{i,j,p,w} = 1 \qquad \forall p, w$$

C2: Each team plays once per week

$$\sum_{j\neq t}\sum_{p} M_{t,j,p,w} + \sum_{i\neq t}\sum_{p} M_{i,t,p,w} = 1 \qquad \forall t, w$$

C3: Each pair plays once

$$\sum_{w}\sum_{p} (M_{i,j,p,w} + M_{j,i,p,w}) = 1 \qquad \forall i < j$$

C4: Period usage limit

$$\sum_{w}\left(\sum_{j\neq t} M_{t,j,p,w} + \sum_{i\neq t} M_{i,t,p,w}\right) \leq 2 \qquad \forall t, p$$

4.2.1 Symmetry breaking

Fix the first week to reduce symmetry:

$$M_{p,n-p-1,p,0} = 1 \qquad \forall p \in \{0,\ldots,P-1\}$$

4.3 Validation

4.3.1 Experimental design

Implementation: PuLP 2.7.0 with CBC solver (version 2.10.3)

Configuration: 300-second timeout, single-threaded

All experiments were conducted on a MacBook Air equipped with an Apple M2 processor and 8 GB of RAM, running macOS, using single-threaded execution.

4.3.2 Experimental results

The MIP approach using the open-source CBC solver is able to solve small instances within the imposed time limit, successfully finding feasible solutions for instances up to $n = 12$. However, the performance of the solver degrades rapidly as the size of the problem increases, and no solution is obtained for

n	**Time (s)**	**Status**
4	300	UNSAT
6	**¡ 1**	SAT
8	**¡ 1**	SAT
10	**1**	SAT
12	**114**	SAT
14	300	Timeout
16	300	Timeout
18	300	Timeout
20	300	Timeout

Table 3: MIP decision results obtained with CBC.

instances with $n \geq 14$ within the 300-second time limit. These results indicate a limited scalability of the MIP formulation when solved with CBC in larger instances.

5 Comparison and Discussion

5.1 Performance comparison

Table 4 compares the three approaches on the decision version.

Instance	**CP (Gecode)**	**CP (Chuffed)**	**SAT (Z3)**	**MIP (CBC)**
n=4	UNSAT	UNSAT	UNSAT	UNSAT
n=6	**¡ 1s**	**¡ 1s**	**¡ 1s**	**¡ 1s**
n=8	**¡ 1s**	**¡ 1s**	**¡ 1s**	**¡ 1s**
n=10	*timeout*	**¡ 1s**	**3s**	**1s**
n=12	*timeout*	**2s**	**45s**	**114s**
n=14	*timeout*	*timeout*	*timeout*	*timeout*
n=16	*timeout*	*timeout*	*timeout*	*timeout*
n=18	*timeout*	*timeout*	*timeout*	*timeout*
n=20	*timeout*	*timeout*	*timeout*	*timeout*

Table 4: Runtime comparison for the decision version.

5.2 Scalability analysis

CP (Gecode) The CP model solved with Gecode scales only to small instances. While instances with $n \leq 8$ are solved almost instantly, Gecode fails to find solutions for $n \geq 10$ within the 300-second time limit. This indicates that, despite the use of symmetry-breaking constraints, the propagation and search strategies employed by Gecode are insufficient to cope with the rapid growth of the search space as the problem size increases.

CP (Chuffed) Chuffed exhibits noticeably better scalability than Gecode on the same CP model. It is able to solve instances up to $n = 12$ within a few seconds, demonstrating the benefit of lazy clause generation and clause learning in pruning the search space. However, this advantage diminishes for larger instances, and Chuffed also fails to solve instances with $n \geq 14$ within the time limit. Overall, while Chuffed extends the practical solvable range of the CP model, scalability remains limited for larger problem sizes.

SAT (Z3) The SAT-based model solved with Z3 shows moderate scalability. Small instances up to $n = 8$ are solved almost instantly, while instances with $n = 10$ and $n = 12$ require several seconds to tens of seconds. Beyond this point, scalability degrades sharply: for $n \geq 14$, Z3 is unable to find a solution within the 300-second time limit. This behavior indicates that, although the SAT encoding is effective for small instances, the search space grows too large for Z3's SAT-based reasoning to handle efficiently at larger scales.

MIP (CBC) The MIP formulation solved with the open-source CBC solver performs well on small instances, solving cases up to $n = 8$ within negligible time. Instances with $n = 10$ and $n = 12$ are also solved, but with increasing runtimes, reaching over one minute for $n = 12$. For larger instances ($n \geq 14$), CBC fails to find a feasible solution within the 300-second time limit. These results highlight the limited scalability of the MIP approach with CBC for larger problem sizes.

5.3 Model complexity

Table 5 presents a runtime comparison for the decision version of the problem across CP (Gecode and Chuffed), SAT (Z3), and MIP (CBC).

Metric	**CP**	**SAT**	**MIP**
Variables ($n = 10$)	90	4,050	4,050
Constraints ($n = 10$)	4 global constraint families	231 PB constraints	231 linear constraints
Model size	Compact	Large	Large
Encoding effort	Low	Medium	Medium

Table 5: Model complexity comparison for $n = 10$.

Table 5 highlights the differences in modeling complexity across the three paradigms for (n=10). The CP model is the most compact, using a small number of variables and a limited set of high-level constraint families, which results in lower encoding effort. In contrast, both SAT and MIP require a much larger number of variables and constraints, leading to larger model sizes and more complex encodings, even though all models represent the same problem.

5.4 Symmetry breaking effectiveness

All three approaches use symmetry breaking, but with different levels of aggressiveness:

- **CP**: Fixes entire first week + home/away orientation for team 1 (most aggressive)
- **SAT**: No explicit symmetry breaking in current implementation
- **MIP**: Fixes first week pattern (moderately aggressive)

The CP model's aggressive symmetry breaking proves most effective, contributing to its superior scalability. The risk of over-constraining is mitigated by careful constraint design ensuring at least one solution always satisfies the symmetry breaking constraints.

5.5 Instance difficulty: n=4

All three approaches correctly identify $n = 4$ as unsatisfiable. Mathematical analysis confirms this:

- With $n = 4$: 3 weeks, 2 periods, each team plays 3 games
- By pigeonhole principle, with only 2 periods available, some team must play in the same period at least $\lceil 3/2 \rceil = 2$ times
- However, the specific constraint structure makes it impossible to satisfy all constraints simultaneously
- This demonstrates the solvers correctly handle infeasible instances

6 Conclusions

In this work, we modeled the Sports Tournament Scheduling problem using three different paradigms: Constraint Programming (CP), propositional Satisfiability (SAT), and Mixed-Integer Programming (MIP), and evaluated them on the decision version of the problem. The experimental study highlights clear differences in modeling compactness and solver scalability across the approaches.

Key findings:

- Within CP, solver choice has a significant impact: while Gecode is effective only on small instances, Chuffed scales further and is able to solve instances up to $n = 12$ within the time limit.

- The SAT approach using Z3 performs well on small instances and remains effective up to $n = 12$, but fails to scale to larger instances.
- The MIP formulation solved with the open-source CBC solver successfully handles small instances and solves cases up to $n = 12$, but does not find solutions for $n \geq 14$ within the imposed time limit.
- All approaches consistently identify the instance with $n = 4$ as infeasible.
- Despite the use of symmetry-breaking constraints, all paradigms encounter a common scalability barrier starting at $n = 14$.
- The results underline the trade-off between modeling compactness and solver performance: CP offers a concise model, while SAT and MIP require substantially larger encodings to represent the same problem.

Authenticity and Author Contribution Statement

We declare that this work is our own and has not been copied from other sources. All ideas and techniques from literature or external resources have been properly cited.

AI tool disclosure: No AI tools were used to generate the CP, SAT, or MIP models, in full compliance with project requirements. The implementations were developed independently by the authors. AI assistance (Claude Code) was used for code organization, documentation, report formatting, and LaTeX template preparation. All modeling decisions, constraint formulations, and experimental analyses are our original work.

Author contributions:

- **Ali Saleh Mohammadabad**: Developed and implemented the MIP model using PuLP with CBC solver. Implemented decision for MIP. Conducted MIP experimental validation. Wrote report section for MIP model and troubleshooting documentation.
- **Saleh Mir Mohammad Rezaei**: Developed and implemented the CP model in MiniZinc, evaluated using both Gecode and Chuffed solvers. Designed and implemented the SAT model using Z3 with Pseudo-Boolean constraints, and wrote the corresponding sections of the report.
- **Both authors**: Collaborative problem formalization, symmetry breaking strategy design, experimental design, comparative analysis, and report integration. Joint development of conclusions and recommendations.

References

[1] Trick, M. (2001). *A Schedule-Then-Break Approach to Sports Timetabling.* In International Conference on the Practice and Theory of Automated Timetabling (pp. 242-253). Springer.

[2] Easton, K., Nemhauser, G., & Trick, M. (2001). *The Traveling Tournament Problem: Description and Benchmarks.* In Principles and Practice of Constraint Programming (pp. 580-584). Springer.

[3] Rasmussen, R. V., & Trick, M. A. (2008). *Round Robin Scheduling – A Survey.* European Journal of Operational Research, 188(3), 617-636.

[4] Gecode Team. (2020). *Gecode: Generic Constraint Development Environment.* Available at: `https://www.gecode.org`

[5] de Moura, L., & Bjørner, N. (2008). *Z3: An Efficient SMT Solver.* In International Conference on Tools and Algorithms for the Construction and Analysis of Systems (pp. 337-340). Springer.

[6] Mitchell, S., O'Sullivan, M., & Dunning, I. (2011). *PuLP: A Linear Programming Toolkit for Python.* Available at: `https://coin-or.github.io/pulp/`

[7] Forrest, J., et al. (2020). *CBC (Coin-or Branch and Cut).* COIN-OR Foundation. Available at: `https://github.com/coin-or/Cbc`

[8] Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., & Tack, G. (2007). *MiniZinc: Towards a Standard CP Modelling Language.* In Principles and Practice of Constraint Programming (pp. 529-543). Springer.

[9] Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of Constraint Programming.* Elsevier.