Combinatorial Decision Making And Optimization

Cirone Cono, cono.cirone@studio.unibo.it
Dardini Jacopo, first2.last2@xxxxx.com
Formichella Gio, gio.formichella@studio.unibo.it
Petrozziello Giulio, first2.last2@xxxxx.com

1 Introduction

In this project we tackle the sport scheduling problem with CP, SAT, SMT and MIP, and apply with all 4 technologies the common approach described in [1]. All implementations take as input the number of teams in the tournament and start by precomputing the weekly matchups for the round-robin tournament; the matchups are stored in a matrix indexed by weeks and periods. The precomputation cost is minimal, for all tested team sizes the precomputation required less than a second. The matchup matrix not only speeds up the search by reducing the problem to ordering matchups, but also guarantees that the constraints of each team playing each other and playing once a week are satisfied. With all four techniques, we aim to find a solution satisfying the constraints and minimizing the objective variable D, the maximum difference between games played at home and away for a team; $D \in [1, n-1]$, where n is the number of teams.

2 CP Model

2.1 Decision Variables

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

2.1.1 matches

The variable $\text{matches}_{p,w} \in [0, \ldots, \frac{N}{2} - 1]$ for $p \in P, w \in W$, determines which match from the predefined round-robin structure (rb) is scheduled in week w and period p.

2.1.2 home_away

The binary variable $\text{home\_away}_{p,w} \in \{0, 1\}$ for $p \in P, w \in W$, assigns the home team for the match scheduled at period p in week w. Specifically, $\text{home\_away}_{p,w} = 0$ means the first team listed in the match plays at home, while $\text{home\_away}_{p,w} = 1$ means the second team plays at home.

2.2 Auxiliary Variables

2.2.1 home_games

The auxiliary variable $\text{home\_games}_t \in [1, \ldots, N - 1]$ for $t \in T$, represents 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 home_away.

2.3 Objective Function

The model's objective is to quantify and minimize disparities in home game assignments through the max_imbalance variable.

2.3.1 Objective Variable: max_imbalance

The integer variable $\text{max\_imbalance} \in [1, \ldots, N - 1]$ quantifies the maximum absolute disparity in home game assignments across all teams. The lower bound of 1 acknowledges that perfect balance ($\text{max\_imbalance} = 0$) is not achievable for an odd number of total games ($N - 1$ games). 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 the following fairness constraint:

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

This formulation precisely defines the absolute "imbalance" for each team t. This imbalance is derived from the difference between a team's home games and away games. Since $\text{home\_games}_t + \text{away\_games}_t = N - 1$ (total games), substituting the expression for away_games_t yields the imbalance for team t as $2 \times \text{home\_games}_t - (N - 1)$. By enforcing that the absolute value of this imbalance for every team must be less than or equal to max_imbalance, this variable effectively captures the largest such deviation among all teams, serving as the direct measure of the overall schedule's fairness.

2.3.2 Objective

The objective is to **minimize max_imbalance**:

$$\text{minimize max\_imbalance}$$

This aims to achieve the fairest possible distribution of home and away games. An optimal solution would ideally yield $\text{max\_imbalance} = 1$, meaning each team's home/away game count differs by at most one, which is the best possible outcome for an odd number of total games played.

2.4 Constraints

2.4.1 Core Constraints

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

1. **Each period must be used exactly once per week:** Ensures that for every week, all matches generated by the round-robin structure's periods are indeed scheduled. Without this, some pairings might be missed, or periods might be duplicated, leading to an incomplete or invalid schedule.

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

2. **Each team plays at most twice per period:**

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

3. **Calculation of Home Games:** This constraint defines the value of home_games_t.

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

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 $\text{home\_away}_{p,w}$.

4. **Fairness - balance home and away games:** This constraint directly links the calculated home_games for each team to the objective variable max_imbalance. For a detailed explanation of this relationship and the definition of max_imbalance, please refer to Section 2.3.1 (Objective Variable and Objective Function).

2.4.2 Implied Constraints

1. **Each team appears exactly once per week:**

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

2.4.3 Symmetry Breaking Constraints

1. **Break period assignment symmetry using lexicographic ordering:** The order in which matches corresponding to P are assigned within matches for each week is symmetrical. This constraint breaks such symmetries by enforcing a lexicographical ordering, reducing the number of equivalent search paths.

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

   This states that the sequence of matches variables, when read in normal (p, w) order, must be lexicographically greater than or equal to when read in (p, w) order with p reversed. This helps to fix one permutation of period assignments.

2. **Fix first match home assignment to break home/away symmetry:** This constraint eliminates global home/away assignment symmetry by fixing the home/away status of the first match.

$$\mathrm{home\_away}_{0,0} = 0$$

3. **Balance the home/away assignments within each week:** While not a strict symmetry breaking constraint, this constraint helps reduce the search space by ensuring that within each week w, the number of matches where the second team plays at home ($\mathrm{home\_away}_{p,w} = 1$) is roughly half of the total matches ($|P|$), with a maximum deviation of 1:

$$\forall w \in W : \left| \sum_{p \in P} \mathrm{home\_away}_{p,w} - \left\lfloor \frac{|P|}{2} \right\rfloor \right| \leq 1$$

   This guides the solver towards balanced assignments and prunes highly imbalanced weekly configurations.

2.5 Validation

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.

2.5.1 Experimental Design

To comprehensively evaluate the performance of different solving strategies for the Sports Tournament Scheduling problem, a systematic experimental study was conducted.

Hardware and Software: Experiments were executed on a MacBook Air M1 equipped with an 8-core CPU. The following solvers were employed: *Gecode*, *Chuffed* and *OR-Tools CP-SAT*. A uniform time limit of 300 seconds was imposed for each individual problem instance.

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

Search Strategies: Three distinct search strategies were employed to analyze solver behavior, with a particular focus on how they influenced Gecode, often considered to have weaker default heuristics compared to modern SAT-based solvers.

Search Strategies: Three distinct search strategies were employed to analyze solver behavior, focusing on their influence on Gecode, given its often weaker default heuristics compared to modern SAT-based solvers.

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`) was applied, prioritizing `matches` variables with `dom_w_deg` and `home_away` variables with `first_fail`, utilizing a `restart_luby(100)` policy.

3. **Relax-and-Reconstruct (LNS) Strategy:** This higher-level strategy incorporated `relax_and_reconstruct` on the `matches` variables (preserving 60% of solution values), leveraging Large Neighborhood Search (LNS) techniques. 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 custom Sequential Custom Search and Relax-and-Reconstruct strategies. This allowed direct evaluation of the user-defined MiniZinc search annotations, rather than the solvers' highly optimized default heuristics.

2.5.2 Experimental Results

n	GECODE				CHUFFED				CP-SAT			
	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB
6	0	0	0	0	0	0	0	0	0	0	0	0
8	6	0	0	7	0	0	0	0	0	0	0	0
10	N/A	N/A	0	N/A	0	0	0	0	1	1	0	1
12	N/A	N/A	0	N/A	5	1	3	2	3	2	2	2
14	N/A	N/A	N/A	N/A	180	21	69	141	5	6	5	5
16	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	16	29	29	13
18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	265	35	39	270
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	84	67	66	82
22	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	238	157	118	253

Table 1: CPU time in seconds for finding the *optimal solution* 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	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	1	1
12	0	0	0	0	0	0	0	0	54	55	58	47
14	4	8	4	7	N/A	57	34	N/A	N/A	N/A	N/A	N/A
16	N/A	N/A	191	125	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 2: CPU time in seconds for finding the *optimal solution* using *Sequential Custom Search Strategy*

n	GECODE				CHUFFED				CP-SAT			
	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB	bs	complete	noIMPL	noSB
6	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1	1	1	1
12	0	0	0	0	8	96	3	2	54	63	59	50
14	1	5	1	0	N/A	N/A	184	250	N/A	N/A	N/A	N/A
16	183	19	1	8	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
18	N/A	3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 3: CPU time in seconds for finding the *optimal solution* using *Relax-and-Reconstruct (LNS) Strategy*

3 MIP

3.1 Decision variables

3.1.1 matches

The binary decision variables X_{wpm} are equal to 1 iff in week w and period p match m is played.

3.1.2 slots

The binary decision variables A_{wp} determine which team plays at home and which away. A is indexed by weeks and periods, so A_wp corresponds to the matchup between teams t_1, t_2 in week w and period p; if $A_{wp} = 0$ then t_1 plays at home and t_2 away, if, otherwise, $A_{wp} = 1$ the order is reversed.

3.1.3 team periods of play

The binary variables TP are used to constrain each team to playing at most twice in the same period. $TP_{twp} = 1$ iff team t plays in week w and period p.

3.1.4 home games counter

To compute the common objective function in MIP, it is necessary to introduce an array of integer auxiliary variables H such that $H[t]$ is the number of home games team t plays, bounded in $[0, n-1]$.

3.2 Objective variables

To compute D as the maximum difference between games played at home and away for all teams, we first find for each team the number of games played at home (2), and then constraint 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. The absolute value of the difference is not computed explicitly but decomposed into 2 inequalities (3)(4). Finally, we look for the minimum of D (1)

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

$$H_t = \sum_{w,p} (A_{wp} = 0 \wedge G_{wp}[\text{home}] = t) + \sum_{w,p} (A_{wp} = 1 \wedge G_{wp}[\text{away}] = t) \qquad t = 1, \dots, n \tag{2}$$

$$D \geq 2H_t - (n-1) \qquad t = 1, \dots, n \tag{3}$$

$$D \geq -(2H_t - (n-1)) \qquad t = 1, \dots, n \tag{4}$$

3.3 Constraints

3.3.1 periods and matches

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).

$$\sum_{m=1}^{n/2} X_{wpm} = 1 \qquad p = 1, \ldots, n/2 \quad w = 1, \ldots, n-1 \tag{5}$$

$$\sum_{p=1}^{n/2} X_{wpm} = 1 \qquad m = 1, \ldots, n/2 \quad w = 1, \ldots, n-1 \tag{6}$$

3.3.2 team playing at most twice in the same period

The constraint on teams playing at most twice in the same period was imposed by first linking the variables of X to those in TP based on the values in G (7) and then imposing that the sum of periods of play is smaller than 2 (8).

$$TP_{twp} = X_{wpm} \qquad G_{wm} = (t1, t2) \wedge (t = t1 \vee t = t2) \qquad \forall t \forall p \forall w \tag{7}$$

$$\sum_{w=1}^{n-1} TP_{twp} \leq 2 \qquad t = 1, \ldots, n \quad p = 1, \ldots, n/2 \tag{8}$$

3.4 Validation

Experimental design

The model was written in Python by making use of the PuLP library and the solvers tested on the MIP model were: CBC 2.10.3, HiGHS 1.10.0, CPLEX 22.1.1 and SCIP 5.5.0 with their default parameters. The time elapsed to find an optimal solution, within the 300 second time limit, was measured and results presented in Fig.1 . All tests were run on a single core of an Intel i7-10750H CPU.

Experimental results

As shown in Fig. 1 CBC had the worst performance, it isn't able to find the optimal solution for n greater than 14. CPLEX, instead, was the fastest up to n=14 but after n=16 it stopped finding the optimal solution. HiGHS was able to find an optimal solution up to n=18 and SCIP, the best performer of the four, was able to reach n=20.

It was also verified that the solvers either found and optimal solution or no solution at all.

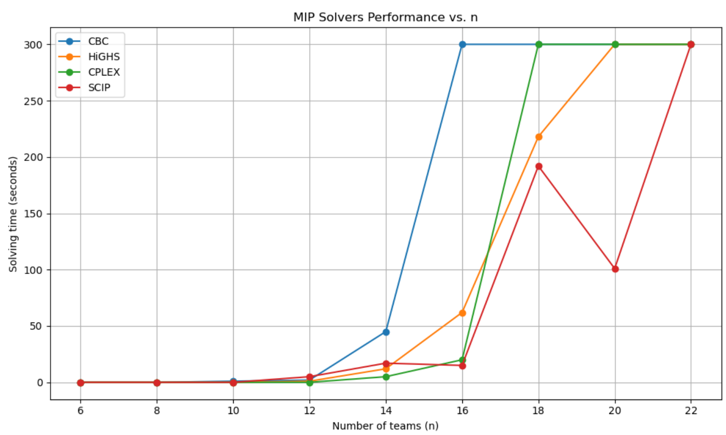

Figure 1: MIP optimization

4 SAT Model

4.1 Decision variables

Let rb be the round robin tournament, sts a schedule satisfying to the STS problem, and P, W, T the set of periods, weeks and teams respectively. The satisfiability task is formalized by the following categories of propositions:

- $matches\_schedule_{p,w,m} \leftrightarrow$ match m, denoting $rb_{m,w} = (t_1, t_2)$, takes place in period p of week w
- $matches\_to\_periods_{t_1,t_2,p} \leftrightarrow t_1$ plays against t_2 in period p

The optimization task, on the other hand, is expressed as follows.

- $slots\_schedule_{p,w} \leftrightarrow$ team $sts_{p,w,1} = t_1$ plays away
- $matches\_to\_slots_{t_1,t_2} \leftrightarrow t_1$ plays away and t_2 plays at home

4.2 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 T be the set of teams and A_t the number of times the team $t \in T$ plays away, the objective function translates into the following:

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

The chosen CP model allows to solve the optimization task independetly of the STS constraints 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 result is encoded in $slots\_schedule_{p,w}$, which cannot express the constraints alone, due to structural limitations of the encoding. Instead $matches\_to\_slots_{t_1,t_2}$ is used: given a week w and a period p, the team $sts_{p,w,1} = t_1$ plays away if, and only if, the team $t_2 = sts_{p,w,2}$ plays at home. Therefore, by definition:

$$\forall p \in P, w \in W.(slots\_schedule_{p,w} \leftrightarrow matches\_to\_slots_{sts_{p,w,1}, sts_{p,w,2}})$$

Finally the optimization process is implemented by ensuring that for an instance $k \in [1, \lfloor \frac{N-1}{2} \rfloor]$:

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

4.3 Constraints

4.3.1 Every team plays once a week

In a Round-Robin tournament rb, each team plays exactly once a week. Therefore the only problem is to ensure that for each week w, every match $rb_{m,w} = (t_i, t_j)$ is assigned to exactly one period p.

We enforce that every match is scheduled once:

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

Finally no two matches are scheduled in the same period.

We ensure that each match is scheduled to a unique period p in the week w.

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

4.3.2 Every team plays at most twice in the same period

The constraint cannot be expressed using directly $matches\_schedule_{p,w,m}$, due to structural limitations of the encoding. Instead, we introduce the literal $matches\_to\_periods_{t_i,t_j,p}$: given a week w, the match $rb_{m,w}$ is scheduled in period p if, and only if, the match $(rb_{m,w,1}, rb_{m,w,2}) = (t_1, t_2)$ takes place in period p. Therefore, by definition:

$$\forall p \in P, w \in W, m \in P.matches\_schedule_{p,w,m} \leftrightarrow matches\_to\_periods_{rb_{m,w,1},rb_{m,w,2},p}$$

Computationally this constraint soundly maps $matches\_schedule$ to $matches\_to\_periods$, which allow to express the main constraint:

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

4.4 Validation

4.4.1 Experimental design

The model was written in Python by making use of the Z3 and the CVC5 library, which offers CaDiCaL and MiniSat as the underlying SAT solvers. The time elapsed to find an optimal solution, within the 300 second time limit, was measured and results presented in Fig.2 .

4.4.2 Experimental results

All solvers are able to find the optimal solution, which was much easier to find once the satisfying one was computed, but overall Z3 had the best performance in time and maximal size of the problem, i.e. $N = 20$. On the other hand CaDiCaL was the worst, failing at $N = 12$, while MiniSat stopped at $N = 14$

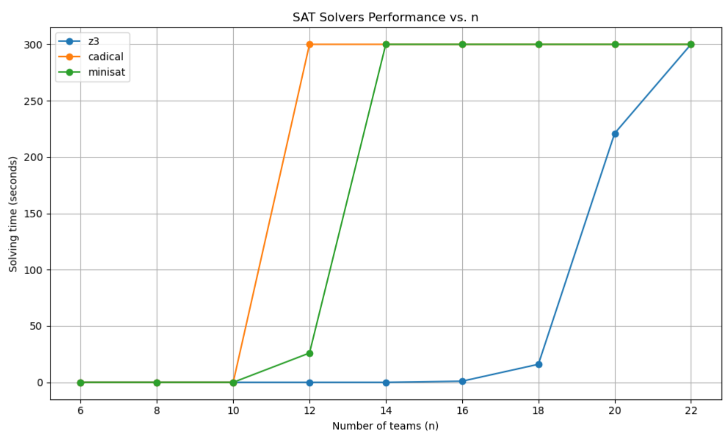

Figure 2: SAT optimization

5 SMT

This section describes our SMT approach for the Sports Tournament Scheduling (STS) problem, modeled using the Quantifier-Free Linear Integer Arithmetic (QF_LIA) theory and solved with Z3 and CVC5. The approach is divided into two phases.

In **Phase 1**, we generate a feasible schedule by selecting a valid assignment of precomputed matches to each slot. Weekly matches are determined in advance using the standard round-robin method. The feasibility constraints ensure that each slot contains exactly one valid pair, each pair is used exactly once, and every team respects participation limits.

In **Phase 2**, we optimize the feasible solution by balancing the number of home and away games for each team. This is done by introducing flip variables that can invert the home and away teams in each match to reduce imbalance. Each phase is encoded in a separate `.smt2` file: the first finds a valid solution, the second refines it by minimizing imbalance.

5.1 Decision Variables

Phase 1:

- $home_{p,w} \in \{1, \ldots, n\}$: team playing at home in period p, week w.
- $away_{p,w} \in \{1, \ldots, n\}$: team playing away in period p, week w.
- $index_{p,w,t} \in \{\text{true}, \text{false}\}$: true if pair t is assigned to slot (p, w).

Phase 2:

- $flip_{p,w} \in \{\text{true}, \text{false}\}$: true if the home and away teams in slot (p, w) are swapped to improve balance.
- $homeEff_{p,w}$, $awayEff_{p,w}$: effective home and away teams after applying flips.

5.2 Objective Function

The objective function applies only to **Phase 2** and aims to minimize the maximum imbalance k between home and away games for any team:

$$k^* = \min\{k : \forall t,\ |H_t - A_t| \leq k\}, \quad H_t = \sum_{p,w}[homeEff_{p,w} = t], \quad A_t = \sum_{p,w}[awayEff_{p,w} = t].$$

The effective assignments are the following.

$$homeEff_{p,w} = \text{ite}(flip_{p,w},\ away_{p,w},\ home_{p,w}), \quad awayEff_{p,w} = \text{ite}(flip_{p,w},\ home_{p,w},\ away_{p,w}).$$

The optimal k^* is found using a binary search over feasible imbalance bounds.

5.3 Constraints

Phase 1:

- **Pair selection:** each slot (p, w) selects exactly one valid pair:

$$\bigvee_t \Big(index_{p,w,t} \wedge home_{p,w} = h_{t,w} \wedge away_{p,w} = a_{t,w} \Big).$$

- **Unique pair usage:** each pair appears exactly once:

$$\sum_p index_{p,w,t} = 1, \quad \sum_w index_{p,w,t} = 1.$$

- **Period participation limit:** each team appears at most twice in the same period over the tournament.

- **Symmetry breaking:** the first pair is fixed: $index_{0,0,0} = \text{true}$.

Phase 2:

- **Balance condition:**

$$|H_t - A_t| \leq k, \quad \forall t.$$

Note: all feasibility constraints from Phase 1 remain implicitly valid, since Phase 2 only modifies the home/away role through flips but does not change the original match assignments.

5.4 Validation

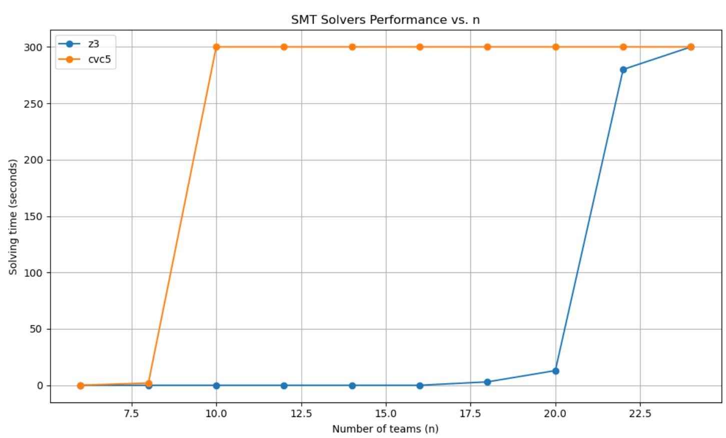

Figure 3:

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.