CDMO Project Report

Katia Gramaccini - katia.gramaccini@studio.unibo.it
Giada Triulzi - giada.triulzi@studio.unibo.it

1 Introduction

In this report, we present various modeling approaches to solve the Sport Tournament Scheduling problem. Our project explores the task by using multiple paradigms: Constraint Programming (CP), Propositional SATisfiability (SAT), Satisfiability Modulo Theories (SMT), Mixed-Integer Linear Programming (MIP) and optimization techniques, applied to CP, SMT and MIP only. For each formalism, we first designed the model and then conducted an experimental study. The experimental evaluation was progressively extended to consider an increasing number of teams, as long as an optimal or suboptimal solution could be computed within the time limit of 5 minutes. In our tabular results, we excluded tournaments with $n = 2$ (trivial case) and $n = 4$ (UNSAT with the given constraints). Therefore, reported experiments start from $n = 6$, up to a maximum size determined by the computational scalability of each individual approach. All models share these common definitions: variables representing $team\_1$ and $team\_2$ are constrained to assume values between 1 and n, *weeks* from 1 to $n - 1$ and *periods* between 1 and $n/2$.

2 CP Model

Four different models were defined, which share the same decision variables but implement different constraints and search strategies. The `basic` model consists of the given constraints plus the implied one. `Local symbreak` and `global symbreak` also include various symmetry breaking constraints, in addition to the ones of the previous model. More details are provided in Section 2.3. Lastly, with the `local noimplied` model we demonstrated that the implied constraint is not redundant.

2.1 Decision variables

Since we want to assign which team plays at home and which plays away in every time slot, inspired by [11], we defined the decision variables `tHome` and `tAway`, each with domain $[1..n]$, common to all models. Both are two dimensional: `tHome`$[i, j] = t$ means that team t plays at home in period i during the j-th week, while `tAway`$[i, j] = t$ means that team t plays away in period i during the

j-th week. For optimization, we introduced the integer variable `totDistance`, which we will be better described in the section that follows.

2.2 Objective function

`totDistance` will be used in the `global symbreak` model to store and minimize the following function:

$$\sum_{t=1}^{n} abs\left(\sum_{i,j} \delta^{h}_{(i,j)}(t) - \sum_{i,j} \delta^{a}_{(i,j)}(t)\right)$$

where $i \in \{1, \ldots, periods\}$ and $j \in \{1, \ldots, weeks\}$
$\delta^{h}_{(i,j)}$ is such that:

$$\delta^{h}_{(i,j)}(t) = \begin{cases} 1 & if \quad \texttt{tHome}[i,j] = t \\ 0 & otherwise \end{cases}$$

$\delta^{a}_{(i,j)}$ is defined analogously for `tAway`. In practice, we count how many times a team has played at home and away, and compute the difference between these two quantities. Then, we store the sum of this value across all teams in `totDistance`. Our objective function will be $min($`totDistance`$)$.

2.3 Constraints

We can now define the problem constraints, which will be applied to all the considered models.

- Every team plays once a week: for every week $j \in \{1, \ldots, weeks\}$, we impose that, for each team, the `global_cardinality` of the values $\{1, \ldots, n\}$ in the list of teams playing in week j is equal to 1. This ensures that the value $t \in \{1, \ldots, n\}$ appears only once for every j. This list is obtained by extracting, for all periods i in week j , the values `tHome` and `tAway`, and concatenating them into a single list.

- Every team plays at most twice in the same period: we impose that, for every period $i \in \{1, \ldots, periods\}$, the `global_cardinality` of the values $\{1, \ldots, n\}$ in the list of teams playing in period i is in the range $[0, 2]$. This ensures that each value $t \in \{1, \ldots, n\}$ appears at most twice for each i. The list is obtained as in the previous case, but for i.

- Each team plays against every other team once: for every pair of teams $(t1, t2)$, with $t1 < t2$ (to avoid counting pairs twice and to exclude the case $t1 = t2$) we impose that the `sum` of matches having $t1$ vs $t2$, plus the `sum` of games with $t2$ vs $t1$ is equal to 1. In this way, both home and away configurations are taken into account.

Additionally, when not implicitly enforced by others, we defined a further constraint to prevent a team from playing against itself. Specifically, for all i, j in their domains, the corresponding elements in `tHome` and in `tAway` arrays must be different.

2.3.1 Implied constraints

With the aim of reducing the search space, we included a single implied constraint in all our models, with the exception of `local noimplied`. Since each team has to play against every other team once, the number of matches must be the same as the number of opponents, thus $n-1$ games.

- Each team plays $n-1$ times: we impose that the `global_cardinality` of the values $\{1,\ldots,n\}$ in the list of teams playing in the entire schedule is equal to $n-1$, for each team. The list is obtained by concatenating the two lists with all the array elements.

2.3.2 Symmetry breaking constraints

In the `local symbreak` and `global symbreak` models we broke period symmetries (permutation of periods) by fixing the entire first week as follows:

- Fix first week schedule: we impose that, for all $i \in \{1,\ldots,periods\}$, the match played in week 1, period i is i vs $i+periods$. This means that e.g. for $n=8$, the matches played in the first week are, in order, 1 vs 5, 2 vs 6, 3 vs 7, 4 vs 8.

By fixing these values, we could also break team symmetries (permutation of team indexes). Moreover, in the `local symbreak` and `local noimplied` models, which have been used to solve the decision problem, we can further break match symmetries. Indeed, when we have to find a feasible solution for the decision problem, we do not take into account home or away constraints so, in the same time slot, we may have two symmetric matches. For example, in week 2, period 1 the cases 1 vs 3 or 3 vs 1 are equivalent. This value symmetry was broken as follows:

- Break home-away symmetries: we impose that, for all $i \in \{1,\ldots,periods\}$ and for all $j \in \{1,\ldots,weeks\}$, the index of the team playing at home in time slot $[i,j]$ has to be lexicographically smaller than the index of the team playing away. In the previous example, we could therefore choose match 1 vs 3 and discard 3 vs 1.

In the `global symbreak` model week symmetries (permutation of weeks, variable symmetry) was broken by lexicographically ordering week columns, as suggested in [3]:

- Break week symmetries: we impose that, for all $j \in \{2,\ldots,weeks-1\}$, the array of teams playing at home during week j has to be lexicographically less than or equal to the one of the home teams, playing in week $j+1$. This is done by exploiting the `lex_lesseq` constraint on the arrays extracted by a custom helper function.

This is sufficient to break week symmetries and it's not necessary to consider also teams playing away, because home teams playing in the same week are all distinct. This constraint is not applied to the first week as it is already fixed.

2.4 Validation

In this section we'll compare all the approaches described in the previous paragraphs. Note that optimization was only reported for the `global symbreak` model, as the constraint used in `local symbreak` hindered the home-away optimization.

2.4.1 Experimental design

We ran the experiments using the MiniZinc Python API on Visual Studio Code with Python v3.11 on a PC with a 32GB AMD Ryzen 7 4800HS CPU - unless mentioned the setup is going to be the same for all the experiments in the paper. All the solvers (Gecode, Chuffed) ran in sequential mode, with a time limit of 300 s. Gecode and Chuffed were used considering different search strategies, since the parameters used in Gecode were not supported by Chuffed. Unless specified, the search strategy was applied on the concatenated list of `tHome` and `tAway`. In particular:

- **Gecode:** the variable choice parameter was dom_w_deg, the value assignment parameter was indomain_random and a Luby restart with scale=250 was applied
- **Chuffed:** in `basic` and `local_symbreak` the variable choice parameter was first_fail, the value assignment parameter was indomain_min and a geometric restart with base=2 and scale=50 was applied. Instead, for `global_symbreak` values to search were not specified and a geometric restart with base=1.5 and scale=100 was applied.

2.4.2 Experimental results

ID	basic Gecode	local sb Gecode	local noimp Gecode	global sb Gecode	global sb+opt Gecode
6	0	0	0	0	**6**
8	1	0	0	1	8
10	17	1	1	55	-
12	-	2	217	-	-
14	-	-	-	-	-

Table 1: Results for CP with Gecode

From Table 1 and 2 we can observe that, up to $n = 8$, all models solve the problem efficiently. The best model for the decision problem is `local symbreak`, while Chuffed is the solver that is able to run the biggest number of instances. Also, the benefit linked to the introduction of the implied constraint can be clearly seen with $n = 14$ in `local` model when using Chuffed and for $n = 12$ when moving to Gecode. Also, in the optimization version, for $n > 6$, the solver reached the timeout even if the minimum possible value was already achieved.

ID	basic Chuffed	local sb Chuffed	local noimp Chuffed	global sb Chuffed	global sb+opt Chuffed
6	0	0	0	0	**6**
8	0	0	0	0	8
10	-	5	2	2	10
12	-	2	5	14	-
14	-	11	-	-	-

Table 2: Results for CP with Chuffed

Results proved that comparable results could be reached with both solvers. In the future, to further improve the scalability, we could optimize the model syntax for a specific solver.

3 SAT Model

The Boolean SATisfiability problem belongs to the complexity class of NP-complete problems, which are among the most challenging in the literature. In this project, we used modern SAT solvers **Z3, Minisat22** and **Glucose3**, which have achieved remarkable performances even on large and complex formulas. In particular, we considered and tested two different encodings of the problem, in order to identify the one that could best simplify the solver's performances.

3.1 Decision variables

The first encoding we adopted to formulate the Tournament Scheduling Problem as a constraint satisfaction problem was inspired by a solution to the Nurse Scheduling Problem, developed during the hands-on sessions of the course. Indeed, the Nurse Scheduling Problem shares several analogies with our task: in both cases, the schedule must be created for n subjects (teams or nurses) over a fixed time interval (weeks or days) and under some common constraints. For instance, a week of the tournament is divided into periods, while a working day is divided into shifts. Each team participating in the championship plays at most one game per week, while each nurse can work only one shift per day. In both cases, each slot can be assigned to a single team or nurse. Adapting the definition of Boolean variables proposed in [9] to the Tournament Scheduling Problem, the encoding was modified as $games[x][y][w][p]$, with $x \in \{1, \ldots, n\_teams\}$, $y \in \{1, \ldots, n\_teams\}$, $w \in \{1, \ldots, n\_weeks\}$, and $p \in \{1, \ldots, n\_periods\}$. The Boolean variable $games[x][y][w][p]$ is True if team x plays against team y in week w and period p, False otherwise. In the case x = y (meaning that a team would play a match against itself, which is obviously invalid) a None value is assigned. The second encoding was introduced due to some limitations in the first one, since it requires nested loops over all indices, resulting in a complexity of $O(n^4)$. Indeed, when the number of teams grows, the solver struggles to find a feasible schedule within the time limit. Therefore, we applied the encoding

proposed in [12], which instead exploits only three indices. In particular, we decided to use two sets of Boolean variables, to separately track home and away matches:
$is\_home[x][w][p]$, with $x \in \{1, \ldots, n\_teams\}$, $w \in \{1, \ldots, n\_weeks\}$, and $p \in \{1, \ldots, n\_periods\}$, is set to True if team x plays at home in week w and period p, False otherwise.
$is\_away[x][w][p]$, with $x \in \{1, \ldots, n\_teams\}$, $w \in \{1, \ldots, n\_weeks\}$, and $p \in \{1, \ldots, n\_periods\}$, set to True or False similarly.

3.2 Constraints

Various constraints can be formally expressed using both the variable $games[x][y][w][p]$ and the variables $is\_home[x][w][p]$, $is\_away[x][w][p]$. Given a set of Boolean variables $(x_1, \ldots, x_n)$, various cardinality constraints have been defined using the Naive Pairwise encoding, which will then be used to express the hard requirements of the problem formulation:

- Cardinality constraint `at_least_one`$(x_1, \ldots, x_n)$: ensures that at least one variable of the set is True
- Cardinality constraint `at_most_one`$(x_1, \ldots, x_n)$: limits the number of True variables to be at most one
- Cardinality constraint `exactly_one`$(x_1, \ldots, x_n)$: defined as the combination of the two constraints above, enforces that at least one and at most one variable is True
- Cardinality constraint `at_most_k`$(x_1, \ldots, x_n; k)$: this is a more general constraint, ensuring that in every set of variables at most k variables are True simultaneously.

The final schedule requires to satisfy three core conditions, plus other two constraints that are implicitly entailed:

- Each team plays against each other team exactly once, regardless of which team plays at home or away. This constraint ensures that $\forall (x, y) \in \{1, \ldots, n_{\text{teams}}\}$ and such that $x \neq y$, exactly one match is scheduled between them during the entire tournament. This means that either team x plays at home against team y or viceversa, but not both. This requirement can be formulated using the `exactly_one` constraint.
- Each team plays at most twice in the same period: this specification can be translated in logical formulation using the `at_most_k` constraint, specifying $k = 2$. However, since this constraint slowed down the solver, we explored alternative solutions. In [5], the author presented a repair-based algorithm which, instead of starting from an empty assignment, begins with a conflicting schedule that violates this constraint. The algorithm then iteratively eliminates conflicts by swapping matches within the same

week, to keep the constraint "Each team plays once a week" satisfied. On the other hand, constraint 10 in [6] inspired a different approach: dividing the schedule into two separate blocks, the first going from week 1 to week/2 and the second from the subsequent week up to the last one. Under this assumption, the original constraint can be reformulated to allow at most one match per team and period in each block. Consequently, when considering the whole schedule, each team plays at most once in the first half and at most once in the second half, thus satisfying the requirement of at most 2 matches per period. However, neither of these alternatives led to a significant improvement in solving time as the problem size increased.

- Each team plays once a week, either at home or away: this constraint enforces that each team has exactly one match scheduled for each week, regardless of the period and the location. Again, this requirement can be formulated using the `exactly_one` cardinality constraint.
- For both encodings, the problem definition implies that, for each week and each period, only one match can be scheduled, ensuring no overlapping between games. As a result, the final output must be a matrix of size $teams/2 \times (teams-1)$, where each slot is occupied by `exactly_one` match.
- Since in the second encoding we considered two sets of Boolean variables, representing whether a team plays at home or away in a given week and period, we imposed that a team cannot be both at home and away in the same time slot.

3.2.1 Symmetry breaking constraints

The final schedule of the tournament is represented as a matrix of assignments, where columns (weeks) and rows (periods) are indistinguishable. Therefore, permuting any two weeks or periods will result in an equivalent solution. Following the approach proposed in [3] we imposed lexicographic ordering between columns. Also, since we decided to fix the matches in the first week, the lexicographical ordering of weeks was not total, but applied from the second one and on. Moreover, after observing that the second encoding was much more efficient compared to the first one, we searched for additional symmetries to break within the 3-dimensional matrix model, which contains symmetries across all of them: periods, weeks (already addressed) and teams. Therefore, we imposed a lexicographical ordering also between teams, enforcing $x < y$.

3.3 Validation

The purpose of the validation phase was to assess the performance of various SAT solvers with different encodings of the problem. Initially, the model was implemented using the first encoding and tested with Z3, Minisat22 and Glucose3. Then, the second encoding was tested using only Z3, as SAT specific solvers already provided good performances. Furthermore, we evaluated the

effect of adding symmetry breaking constraints by including them in the second round of experiments, to evaluate their impact on solver efficiency and speed in producing a feasible solution.

3.3.1 Experimental design

The experiments were conducted using the Z3 Python API. A time limit of 5 minutes was imposed, as requested by the problem specifications. No custom search strategy was imposed, solvers used their default decision heuristic.

3.3.2 Experimental results

For each n, we ran the model using 6 different solver configurations for encoding 1 and 2 configurations for encoding 2. Table 3 summarizes the performances obtained for the first encoding.

ID	**Z3 w/out SB**	**Z3 + SB**	**Minisat22 w/out SB**	**Minisat22 + SB**	**Glucose3 w/out SB**	**Glucose3 + SB**
6	0	1	0	0	0	0
8	-	-	0	0	0	0
10	-	-	3	4	11	3

Table 3: **Encoding 1**, results using Z3, Minisat22 and Glucose3 with and without symmetry breaking.

From these results, we observe that the Z3 solver performed poorly with this encoding, since it was only capable of solving the instance with 6 teams involved in the tournament. This behavior may be explained by the fact that Z3 is primarly an SMT solver, which combines SAT solving with additional theories and such generality introduces overhead in the calculation. In this context, the introduction of symmetry breaking constraints did not yield any improvement for Z3. On the other hand, dedicated SAT solvers such as Minisat22 and Glucose3 achieved better performances. Both were capable of solving larger instances, up to 10 teams, within the time limit, even though the introduction of symmetry breaking constraints did not result in a significant speed up. Aiming to improve Z3's performance, the second encoding, which reduces the problem dimensionality, was tested exclusively on it, either with and without symmetry breaking. The corresponding experimental results are shown in Table 4. Thanks to a simplified encoding, Z3 was able to handle larger instances more efficiently. In particular, the introduction of symmetry breaking constraints proved to be effective in the case $n = 12$, reducing the solving time with respect to the standard approach and enabling Z3 to scale effectively to more complex instances.

ID	Z3 w/out SB	Z3 + SB
6	0	0
8	0	0
10	12	2
12	-	18

Table 4: **Encoding 2**, results using Z3 with and without symmetry breaking.

4 SMT Model

Satisfiability Modulo Theory (SMT) determines whether a logical formula is satisfiable with respect to some background theories. Optimization Modulo Theory (OMT) is an extension of SMT, in which the goal is not only to find a satisfiable assignment, but one that also optimizes a given objective function. Z3 solver used in this section natively supports OMT, enabling both constraint satisfaction and optimization within the same framework.

4.1 Decision variables

Similarly to section 2.1, we define two sets of Integer variables, representing the teams involved in each match scheduled in week w and period d:
$home\_team[w][p]$, with $w \in \{1, \ldots, n\_weeks\}$, and $p \in \{1, \ldots, n\_periods\}$
$away\_team[w][p]$, with $w \in \{1, \ldots, n\_weeks\}$, and $p \in \{1, \ldots, n\_periods\}$
An assignment in the form $home\_team[w][p] = x$ means that team x plays at home in week w and period p. The same interpretation holds for the away teams. Each variable is constrained to assume values in the range $\{1, \ldots, n_{\text{teams}}\}$ and it's enforced that a team cannot play against itself: $\texttt{home\_team}[w][p] \neq \texttt{away\_team}[w][p]$, $\forall w, p$. Each pair $(home\_team, away\_team)$ thus represents a valid match between two teams. The theory used is $\mathcal{T}_{\mathcal{Z}}$, namely the Presburger arithmetic.

4.2 Objective function

In the optimization phase, we decided to combine both optimization strategies offered by OMT. For each team, we computed the number of matches scheduled to be played at home during the tournament (`home_count`). Since each team is expected to play $teams - 1$ matches in total, we constrained each value in $home\_count$ to lie within the interval $[min\_home, max\_home]$. The objective is to minimize the overall difference between max_home and min_home, allowing a fairer assignment of home games slots for all teams. Indeed, if all teams play approximately the same number of matches at home, allowing a small tolerance due to the odd total number of matches per team, this difference will be minimized. To further improve fairness, inspired by the work [12], we introduced 3 soft constraints. These are associated with weights and are used to guide the solver towards preferable solutions, without making them strictly mandatory to satisfy:

- Try to avoid consecutive home matches for the same team, with a weight of 10
- Try to avoid consecutive away matches for the same team, with a weight of 10
- We assumed that the last period could represent the weekend and we tried to ensure that all teams have a balanced number of weekend games, with a weight of 2

These additional preferences aim to improve the regularity of the entire tournament schedule. However, since the core fairness is already ensured by balancing home-away games, violations to these soft constraints do not impact the validity of the solution. To better illustrate the fairness achieved by the optimization process, we provide a graphical representation of an optimal schedule for 6 teams. The plot indeed highlights that the difference between the team with the highest number of games played at home and the one with the fewest is minimized.

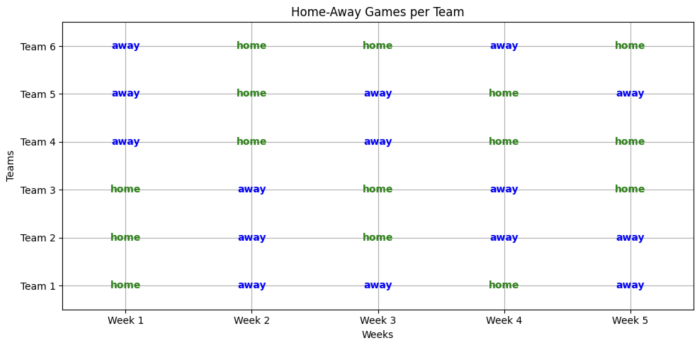

Figure 1: Optimization for $teams = 6$.

4.3 Constraints

We define the following hard constraints required by the problem:

- Each pair of teams plays exactly once: $\forall(x, y)$ with $x < y$, we guarantee that these teams face each other exactly one time during the tournament by summing, over all weeks and periods, the occurrences where home_team[w][p] = x and away_team[w][p] = y, plus the viceversa. The total sum must be equal to 1, ensuring a single match between the two considered teams, regardless of the home or away status.
- Teams play at most twice in the same period: for each team and each period, we count the number of matches in which the team is involved, either at home or away and sum across all weeks. This count must be less than or equal to 2.

- Teams play once a week: for every week, we collect all teams scheduled to play in a given period, either at home or away, and enforce that these teams are all different. This control prevents any team from being scheduled for multiple matches within the same week.

4.3.1 Implied constraints

During the optimization phase, we exploited the fact that each team must play exactly $teams - 1$ matches in total, as the tournament follows a Round Robin format. Although this condition was not explicitly formalized as a constraint in the SMT model, it was a key assumption when defining the range for the optimal number of home games. Specifically, the number of home matches for each team was constrained to lie within the interval min_home: $(teams - 1)//2$ and max_home: $((teams - 1)//2) + 1$.

4.3.2 Symmetry breaking constraints

In the SMT formulation of the problem, two symmetry breaking constraints proved to be particularly effective:

- Fixing the matches in the first week: to eliminate symmetries caused by team permutations, we arbitrarily fixed the matches in the first week. Specifically, for each period in the first week, the home team is assigned to team number p, while the away team is $p + number\_of\_periods$. This choice prevents the solver from finding equivalent solutions that differ only by some swaps in the initial games of the schedule.

- Lexicographical order between columns: for every pair of consecutive columns (weeks), we enforced a lexicographical ordering on the matches scheduled in the respective periods. In particular, for each period p we imposed that the home team in week w must be less than or equal to the home team in week $w+1$. If these two are equal, away teams are compared similarly.

4.4 Validation

The validation phase involved testing the model using the Z3 SMT solver, both with and without the inclusion of symmetry breaking constraints. Subsequently, the solver was switched to the Optmize module and combined with the previously described soft constraints, to balance the distribution of home and away games.

4.4.1 Experimental design

The experiments were conducted using the Z3 Python API. A time limit of 5 minutes was imposed, as requested by the problem specifications. No custom search strategy was imposed, solvers used their default decision heuristic.

For the optimization process, instead of maximizing or minimizing the sum of weights associated to each soft constraint, a custom objective function was specified. As explained in section 4.2, soft constraints were introduced as an additional means to achieve a fairer schedule.

4.4.2 Experimental results

Table 5 summarizes the performance of the solver, in both the decision and optimization phase, for the SMT-based formulation.

ID	Z3 w/out SB	Z3 + SB	Z3 opt w/out SB	Z3 + opt + SB
6	0	0	**1**	**1**
8	4	1	5	5
10	116	38	-	-

Table 5: Results using Z3, with and without symmetry breaking, and optimization.

From these results it's clear that symmetry breaking constraints have a significant impact on the performance. For instance, while the basic version without symmetry breaking required 116 seconds to find a solution for 10 teams, the introduction of symmetry breaking reduced this time to just 38 seconds. However, when increasing the number of teams to 12, which is not shown in the table, the problem becomes considerably more complex and no solution could be found within the time limit. On the other hand, regarding the optimization phase, optimal results were obtained only in the case with $n = 6$. For 8 teams the solver returned a sub-optimal solution and for 10 teams optimization failed to produce any result in the 5 minutes bound.

5 MIP Model

For the MIP formulation, two models are presented: the `base`, which has the core problem constraints, and the `symbreak`, which also includes different symmetry breaking constraints. In both cases, constraints are inspired by [4]. The language used is Pyomo and the solvers are HiGHS and CBC.

5.1 Decision variables

The following decision variables have been defined:

- $x[t1, t2, w, p] \in \{0, 1\}$, which is a binary variable similar to *games*, used for the first SAT encoding in Section 3.1
- $tp[t, p] \in \mathbb{R}_+$, that stores the sum of matches played per team $t \in \{1, \ldots, n\}$ and period $p \in \{1, \ldots, periods\}$
- $abs\_diff[t] \in \mathbb{R}_+$, which stores the quantity that has to be minimized by the optimization function

The first decision variable will be used in all the models, while the second will not be inserted in the optimization version of the `symbreak` model. The last variable, instead, will appear only in the optimization models.

5.2 Objective function

The objective function we want to minimize is the same as the one defined in Section 2.2 for the CP models, but since it is non-linear due to the presence of $abs()$, we first need linearize it. For this purpose, the auxiliary variable abs_diff was introduced, defined in the following way: consider

$$abs\left(\sum_{i,j}\delta^{h}_{(i,j)}(t)-\sum_{i,j}\delta^{a}_{(i,j)}(t)\right)=|\delta(t)| \qquad \forall t\in\{1,\dots,n\}$$

we can define $|\delta(t)| \leq abs\_diff[t]\ \forall t$ and, as proposed in [2], we can use the linearized version with constraints `linearize_abs_diff_1` and `linearize_abs_diff_2` $-abs\_diff[t] \leq \delta(t) \leq abs\_diff[t]\ \forall t$.
Therefore, the objective function will be: $\min\left(\sum_t abs\_diff[t]\right)$.

5.3 Constraints

The core problem constraints share the same definitions in all the proposed models:

- One match per time slot: $\sum_{t1,t2} x[t1,t2,w,p]=1 \quad \forall w,p$
- Each team plays against every other team once: $\sum_{w,p} x[t1,t2,w,p] + x[t2,t1,w,p]=1 \quad \forall t1,t2 : t1<t2$
- Each team cannot play against itself: $x[t,t,w,p]=0 \quad \forall t,w,p$
- Every team plays once a week: $\sum_{p,t2\neq t1} x[t1,t2,w,p]+x[t2,t1,w,p] = 1 \quad \forall w,t1$

Unless specified differently, the domains for each index are: $w\in\{1,\dots,weeks\}, p\in\{1,\dots,periods\}, t,t1,t2\in\{1,\dots,n\}$. We also have to ensure that each team plays at most twice in the same period. For this purpose, we define the following channeling constraint:

$$\sum_{w,t2\neq t1} x[t1,t2,w,p]+x[t2,t1,w,p]=tp[t1,p] \quad \forall p,t1$$

and we impose that $tp[t1,p]\leq 2 \quad \forall t,p$. This constraint is applied to all models, with the exception of the optimization version of `symbreak` in which, with the same scope, we imposed:

$$\sum_{w,t2\neq t1} x[t1,t2,w,p]+x[t2,t1,w,p]\leq 2 \quad \forall p,t1$$

5.3.1 Symmetry breaking constraints

In order to break multiple kinds of symmetries at the same time in the `symbreak` models, we extracted a sub-matrix from the schedule considering the first $\frac{n}{2}$ weeks and fix the matches along its diagonal by imposing that, for all $p \in \{1, \ldots, periods\}$, $x[2p-1, 2p, p, p] = 1$. This means that, for example, with $n = 6$ we can fix the matches 1 vs 2 in week 1, period 1, 3 vs 4 in week 2, period 2 and 5 vs 6 in week 3, period 3. By imposing these matches, we forbid period and team indexes permutations. Fixing them along the diagonal implies that week permutations involving the first $\frac{n}{2}$ are excluded from the search.

5.4 Validation

We now present the experimental results obtained with our models.

5.4.1 Experimental design

We ran the experiments using the Pyomo software. All the solvers (HiGHS, CBC) ran in sequential mode with a time limit of 300 s. Search heuristics cannot be imposed here, and the only parameter set for solvers was $threads = 1$, to ensure sequentiality.

5.4.2 Experimental results

ID	base CBC	base HiGHS	sb CBC	sb HiGHS	base+opt CBC	base+opt HiGHS	sb+opt CBC	sb+opt HiGHS
6	0	0	0	0	6	**6**	**6**	**6**
8	0	3	0	0	8	**8**	8	**8**
10	2	12	0	7	10	**10**	10	**10**
12	94	-	32	242	12	-	24	-

Table 6: Results for MIP, with and without symmetry breaking.

From Table 6 we can observe that a feasible schedule is found by all the models up to $n = 10$, with CBC performing consistently better than HiGHS in terms of scalability for the decision problem. Note that HiGHS either stops when the optimal value for the objective function is reached or it does not even find a suboptimal solution. Symmetry breaking constraints lead to an improvement in the performance for the decision version, allowing HiGHS to find a solution also in the case with $n = 12$. On the other hand, they seem to hinder the optimization problem.

6 Conclusions

In this report, we have presented and compared several approaches to solve the Tournament Scheduling Problem. Our goal was to explore as many problem for-

mulations and techniques as possible among those introduced during the course, in order to better understand how to turn theoretical concepts into practical applications, evaluate the different performances and identify the most effective strategy for this specific task. Among the tested models, the combination of Constraint Programming with the Chuffed solver and symmetry breaking constraints proved to be the most efficient, successfully solving instances up to 14 teams in just 11 seconds. On the contrary, most of the other approaches were only able to handle tournaments involving a maximum of 12 teams, within the same time limit. It's worth noting that even when a solver performs well for a certain problem size, returning a suitable schedule in just a few seconds, it may struggle as the problem scales. For instance, although Z3 combined with symmetry breaking constraints was efficient for 10 teams in the SMT formulation, it failed to find a feasible schedule, satisfying the time limit, when the number of teams was increased to 12. This highlights how both the choice of the solver and the integration of specific constraints can significantly influence performance and scalability in real-world scenarios.

Authenticity and Author Contribution Statement

We declare that the work described in this report is our own, except where explicitly stated otherwise, and that we have not copied it from someone else. All external ideas, concepts or materials taken from any person or resource have been properly cited within the text. Relevant authors' contribution to the work are listed in the reference section below. We used the AI tool ChatGPT to improve the clarity and correctness of some explanations in the report and to assist in the translation of mathematical formulas into the LaTeX format. Additionally, we employed ChatGPT to assist in configuring the Dockerfile and in writing the required scripts for the creation of the JSON format output and for compiling all the models together or a specific model on the desired number of teams.

References

[1] Dirk Briskorn and Andreas Drexl. "IP models for round robin tournaments". In: *Computers Operations Research* 36.3 (2009), pp. 837–852. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2007.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0305054807002298.

[2] M. Chan et al. *Optimization with absolute values*. https://optimization.cbe.cornell.edu/index.php?title=Optimization_with_absolute_values. 2025.

[3] Pierre Flener et al. “Breaking Row and Column Symmetries in Matrix Models”. In: *Principles and Practice of Constraint Programming - CP 2002*. Ed. by Pascal Van Hentenryck. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 462–477. ISBN: 978-3-540-46135-7.

[4] Eva Linda Gunnarsdóttir. “An integer programming formulation for scheduling of the Icelandic football league”. MA thesis. 2019. URL: https://skemman.is/bitstream/1946/33993/1/MSC-EvaLindaGunnarsdo%CC%81ttir_Spring2019_Skemman.pdf.

[5] Jean-Philippe Hamiez and Jin-Kao Hao. “A linear-time algorithm to solve the Sports League Scheduling Problem (prob026 of CSPLib)”. In: *Discrete Applied Mathematics* 143.1 (2004), pp. 252–265. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2003.10.009. URL: https://www.sciencedirect.com/science/article/pii/S0166218X04001350.

[6] Andrei Horbach, Thomas Bartsch, and Dirk Briskorn. “Using a SAT-solver to schedule sports leagues”. In: *J. of Scheduling* 15.1 (Feb. 2012), pp. 117–125. ISSN: 1094-6136. DOI: 10.1007/s10951-010-0194-9. URL: https://doi.org/10.1007/s10951-010-0194-9.

[7] Zeynep Kiziltan and Roberto Amadini. *Combinatorial Decision Making and Optimization - Course Notes*. 2025.

[8] Donald E. Knuth. “Literate Programming”. In: *The Computer Journal* 27.2 (1984), pp. 97–111.

[9] A. Quarta. *CDMO-exercises*. https://github.com/NglQ/CDMO-exercises. 2025.

[10] Toby Walsh. *CSPLib Problem 026: Sports Tournament Scheduling*. Ed. by Christopher Jefferson et al. http://www.csplib.org/Problems/prob026.

[11] xcsp3team. *PyCSP3-models/SportsScheduling*. https://github.com/xcsp3team/PyCSP3-models/blob/main/academic/SportsScheduling. 2025.

[12] Hantao Zhang, Dapeng Li, and Haiou Shen. “A SAT Based Scheduler for Tournament Schedules.” In: Jan. 2004.