Exploration of different Optimization Paradigms for the Sports Tournament Scheduling (STS) problem

Matteo Preda - matteo.preda2@studio.unibo.it
Raffaele Sali - raffaele.sali@studio.unibo.it
Marco Zocco Ramazzo - marco.zoccoramazzo@studio.unibo.it

January 2026

1 Introduction

The report provides an in-depth study of the Sports Tournament Scheduling (STS) problem [1], emphasizing the formulation and comparison of different optimization models. The models examined in this project were built upon a shared formalization, which is outlined in the section 1.1. The main goal of the presented work was to build a unified infrastructure for modeling, analyzing, and solving the STS problem, by considering the different techniques in these areas.

The study was conducted employing two distinct methodological approaches for the construction of the match schedule:

- The schedule was generated from scratch, through the application of fundamental constraints in combination with additional symmetry-breaking and implied constraints. During the whole project, this strategy is defined as the *canonical method*.
- The set of match pairings for each week was determined deterministically via the circle method [2] [3]. Subsequently, the matches within each week were ordered so as to satisfy the period-related constraints and, in the case of an optimized solution, to ensure an appropriate balance between home and away fixtures. A detailed explanation can be found in section 1.2.

1.1 Model Formalization

Input Parameters. The STS problem was defined by the following parameters, which are common to all models:

- n: Number of teams (even number). Teams are indexed by $t \in T = \{1, 2, \ldots, n\}$
- $w = n - 1$: Number of weeks. Weeks are indexed by $w \in W = \{1, 2, \ldots, n - 1\}$
- $p = n/2$: Number of periods (matches per week). Periods are indexed by $p \in P = \{1, 2, \ldots, n/2\}$

Objective Variable and its Bounds. The objective function, which is only related to the optimization version, represents the maximum imbalance in home and away games for any team, with the goal of minimizing it:

$$\min M = \max_{t \in T} |h_t - a_t| \tag{1}$$

where h_t and a_t represent the number of home and away games played by the team t, respectively. The theoretical upper bound for M is $n - 1$, while the lower bound for M is 1, since having an even n implies that the number of matches each team plays are odd. For this reason, h_t and a_t must be different. This objective function has been used in all optimization techniques except MIP, since the absolute value function is nonlinear. In MIP, the objective function has been decomposed into:

$$\min M \quad \text{s.t.} \quad \begin{cases} h_t - a_t \le M, & \forall t \in T, \\ a_t - h_t \le M, & \forall t \in T. \end{cases} \tag{2}$$

Thanks to these two constraints, the absolute value can be represented in a linear form, and the minimization forces M to be equal to $\max_t |h_t - a_t|$.

Constraints. All models obtained with canonical method rest on the following core constraints:

- Every team plays every other team only once:

$$\sum_{w \in W} \sum_{p \in P} x_{ijwp} = 1 \quad \forall i, j \in T,\ i < j$$

- Every team plays once a week:

$$\sum_{j \in T \setminus \{t\}} \sum_{p \in P} x_{tjwp} = 1 \quad \forall t \in T,\ \forall w \in W$$

- Each period in each week has exactly one game:

$$\sum_{t \in T} \sum_{j \in T,\ j > t} x_{tjwp} = 1 \quad \forall w \in W,\ \forall p \in P$$

- Every team plays at most twice in the same period:

$$\sum_{w \in W} \sum_{j \in T \setminus \{i\}} x_{ijwp} \le 2 \quad \forall i \in T,\ \forall p \in P$$

- A team cannot play against itself:

$$x_{ttwp} = 0 \quad \forall t \in T,\ \forall w \in W,\ \forall p \in P$$

Implied Constraint Moreover, the following implied constraint has been used for all the models:

- Total matches per team: Every team plays exactly $n - 1$ matches over the tournament, where $n = |T|$ is the number of teams.

$$\sum_{j \in T \setminus \{t\}} \sum_{w \in W} \sum_{p \in P} x_{tjwp} = n - 1 \quad \forall t \in T$$

Pre-processing steps and Symmetry Breaking constraints. To improve solver performance, the examined models may include pre-processing steps. In particular, the model is built using a solver-independent modeling language such as AMPL [4], which can require additional preprocessing time prior to the actual solving phase.

1.2 Schedule generation

The generation of matches has been studied using two different methods: starting from the more traditional approach of generating weeks and the matches within them from scratch, and moving to the more advanced method of deterministic construction of matches within each week, known as the circle method. The Circle Method (as employed in the context of scheduling) is a classical construction for generating round-robin tournament schedules: teams are arranged around a "circle" and then rotated to determine pairings each round [5].

In the circle method, given n number of teams, the assignment of matches within each week ($n - 1$ in total) is simply based on two rules:

- In week w, team w plays against team n
- For $i, j \in N \setminus \{w, n\}$: team i plays team j if $\text{mod}\left(\frac{i+j}{n-1}\right) \equiv \text{mod}\left(\frac{2w}{n-1}\right)$.

Figure 1 shows a visual representation of the first three weeks generated by the circle method for $n = 8$.

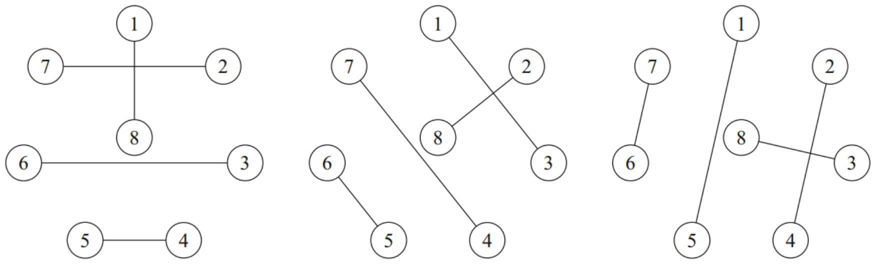

Figure 1: Matches configuration for first three weeks, $n = 8$, according to circle method [5].

Finally, for the reproducibility of the analysis, a random seed was specified for each solver used in the different optimization techinques.

1.3 Code Availability

The project work can be found at `https://github.com/mpreda01/CDMO_project`.

2 CP Model

Constraint Programming (CP) solves combinatorial problems by modeling variables with finite domains and constraints, using domain propagation and search rather than purely logical reasoning or numeric optimization [6] [7]. Both approaches explained in section 1 were implemented in MiniZinc [8] and systematically evaluated across multiple problem sizes through exhaustive experimentation with all possible combinations of symmetry breaking constraints, implied constraints, restart policies, large neighborhood search heuristics, and variable selection strategies.

2.1 Decision variables

2.1.1 Canonical method

The model employs integer variables that encode the matches in each *slot* (w, p). Represented by two parallel arrays:

- *home* $\in \{1, \dots, n\}^{pw}$: the home team assigned to each slot;
- *away* $\in \{1, \dots, n\}^{pw}$: the away team assigned to the same slot.

Auxiliary variables

- `period_count[t, p]` $\in \{0, 1, 2\}$: Number of appearances of team t in period p, used with `global_cardinality` to limit period occupancy.
- `home_count[t]` $\in W$: Total home games for team t, linked to `home` variables via `global_cardinality`.
- `away_count[t]` $\in W$: Total away games for team t, linked to `away` variables via global_cardinality.
- `team_period[t, w]` $\in P$: Period identifier for team t in week w, enabling symmetry breaking on team scheduling patterns.

2.1.2 Circle method

- $assign \in \{1, \ldots, m\}^{P \times W}$: the match index from the circle method schedule assigned to period p in week w.

Each entry $assign[p, w]$ specifies which match from the predefined set (indexed in *home* and *away*) is scheduled in period p of week w. The core constraint ensures $\{assign[p, w] \mid p \in P\} = games\_in\_week[w]$ for each week w, meaning matches can only be permuted among periods within their designated week, preserving the circle method's structural guarantees.

Auxiliary variables

- $week\_of(i) = ((i - 1) \bmod (n - 1)) + 1$: maps match index i to its designated week (line 30);
- $games\_in\_week[w] = \{i \in \{1, \ldots, m\} \mid week\_of(i) = w\}$: the set of match indices belonging to week w.

2.2 Constraints

2.2.1 Canonical Method

The following core constraints are common to all the solutions, see the mathematical definition in section 1.1, in the CP solution they are implemented with the following global constraints: every team plays with every other team only once is coded with `count`, every team plays once a week is implemented using `all_different`, every team plays at most twice in the same period is implemented with `global_cardinality`.

Symmetry Breaking Constraints

- **Home/away ordering:** $\forall p \in P, w \in W : \text{home}[p, w] < \text{away}[p, w]$ — eliminates equivalent schedules by swapping home/away roles.
- **Week ordering:** $\forall w \in \{1, \ldots, |W| - 1\} : [\text{home}[p, w]]_p \leq_{\text{lex}} [\text{home}[p, w + 1]]_p$ — orders weeks lexicographically by home teams.
- **Period ordering:** $\forall p \in \{1, \ldots, |P| - 1\} : [\text{home}[p, w] \mid \text{away}[p, w]]_w \leq_{\text{lex}} [\text{home}[p + 1, w] \mid \text{away}[p + 1, w]]_w$ - orders periods lexicographically.
- **First week fixing:** $\forall p \in P : \text{home}[p, 1] = p \land \text{away}[p, 1] = n - p + 1$ — fixes team assignments in first week to canonical form.
- **Team 1 period ordering:** $\text{team\_period}[1, 1] = 1 \land \forall w \in \{1, \ldots, n_{\text{weeks}} - 1\} : \text{team\_period}[1, w] \leq \text{team\_period}[1, w + 1]$ — fixes team 1's first period and orders its periods non-decreasingly.

Implied Constraints

- **Period occupancy:** $\forall t \in T : \text{count}(\{\text{period\_count}[t, p] = 0 \mid p \in P\}) \leq 1$ — each team skips at most one period.

2.2.2 Circle Method

General Constraints For the circle method, given the predefined match arrays *home* and *away* generated by the round-robin algorithm, the following constraints are applied:

- **Games cannot change week:** $\{assign[p, w] \mid p \in P\} = \{i \in \{1, \ldots, m\} \mid week\_of(i) = w\}\ \forall w \in W$ using set equality — matches can only be permuted among periods within their week.
- **Period occupancy limit:** $|\{w \in W \mid home[assign[p, w]] = t \lor away[assign[p, w]] = t\}| \leq 2\ \forall p \in P, t \in T$ using `count` — each team plays at most twice per period.

Symmetry Breaking Constraints The circle method employs scale-dependent symmetry breaking strategies:

- **For small instances ($n < 8$) - Period ordering:** $[assign[p, w]]_w \leq_{\text{lex}} [assign[p+1, w]]_w \ \forall p \in \{1, \ldots, |P|-1\}$ — orders periods lexicographically.
- **For medium instances ($8 \leq n < 14$) - Team n fixing:** $home[assign[p_{\text{target}}, w]] = n \vee away[assign[p_{\text{target}}, w]] = n \ \forall w \in W$ where $p_{\text{target}} = ((w-1) \bmod |P|) + 1$ — fixes team n to specific period per week.
- **For large instances ($n \geq 14$) - Team $n-1$ fixing:** $home[assign[p_{\text{next}}, w]] = n-1 \vee away[assign[p_{\text{next}}, w]] = n-1 \ \forall w \in \{1, \ldots, |W|-2\}$ where $p_{\text{next}} = (p_{\text{target}} \bmod |P|) + 1$ — fixes team $n-1$ to adjacent period.

Implied Constraints

- **Match distinctness within weeks:** $\text{all\_different}([assign[p, w]]_p) \ \forall w \in W$ — ensures each period in a week is assigned a different match.

2.3 Validation

The validation framework employs systematic exhaustive testing across all parameter configurations to assess the individual and combined impact of optimization techniques on solver performance. Both the canonical and circle methods systematically evaluate $2^7 = 128$ parameter combinations for each problem size, exploring configurations across seven binary flags: SB (Symmetry Breaking constraints), IC (Implied Constraints), IS (Integer Search), RL (Restart Luby strategy), RR (Relax Reconstruct large neighborhood search), CH (Chuffed solver vs. Gecode [9, 10]), and optimization mode. Each model name in the results represents a specific combination of these enabled solving techniques.

ID	IC+IS +RR	IS	IC+IS +CH	SB+IC +RR+CH
2	0|**0**	0|**0**	0|**0**	0|**0**
4	UNSAT	UNSAT	UNSAT	UNSAT
6	0|**0**	0|**0**	0|**0**	0|**0**
8	0|**0**	0|**0**	1|**1**	0|**0**
10	1|**1**	1|**1**	1|**1**	6|**6**
12	126|**126**	122|**122**	N/A	N/A
14	128|**128**	137|**137**	N/A	N/A
16	N/A	N/A	N/A	N/A
18	N/A	N/A	N/A	N/A
20	N/A	N/A	N/A	N/A

Table 1: Canonical method

ID	RL+RR+CH +SB+IC	RL+CH +SB+IC	IS+RL +RR
2	0|**0**	0|**0**	0|**0**
4	UNSAT	UNSAT	UNSAT
6	0|**0**	0|**0**	0|**0**
8	0|**0**	0|**0**	0|**0**
10	0|**0**	0|**0**	0|**0**
12	1|**1**	1|**1**	1|**1**
14	3|**3**	3|**3**	2|**2**
16	5|**7**	3|**4**	6|**6**
18	15|**17**	3|**4**	N/A
20	187|**189**	189|**190**	N/A

Table 2: Circle method

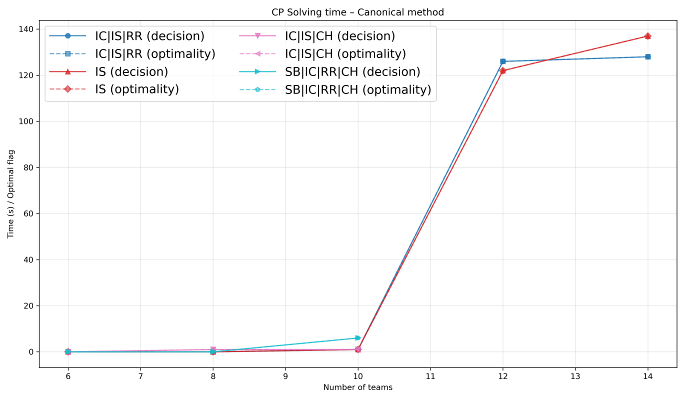

Figure 2: CP solving time – Canonical method with different constraint configurations.

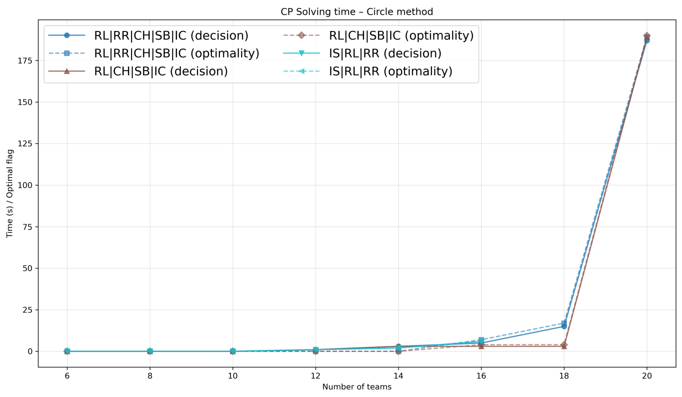

Figure 3: CP solving time – Circle method with different constraint configurations.

3 SAT Model

Boolean Satisfiability (SAT) determines whether there exists an assignment of truth values to Boolean variables that makes a propositional logic formula true [11]. The SAT model was implemented using the Z3 solver [12] and, for solver-independent validation, exported to the DIMACS CNF format [13] to be used with external solvers (MiniSat [14], Glucose [15]).

3.1 Decision Variables

3.1.1 Canonical Method

The core decision variables are 3D Boolean arrays encoding the home/away assignment for each slot:

- $\texttt{home}_{w,p,t} \in \{0,1\}$ where $w \in [0, n-2]$, $p \in [0, \lfloor n/2 \rfloor - 1]$, $t \in [0, n-1]$.
  $\texttt{home}_{w,p,t} = 1$ iff team t plays at home in week w, period p.
- $\texttt{away}_{w,p,t} \in \{0,1\}$ where $w \in [0, n-2]$, $p \in [0, \lfloor n/2 \rfloor - 1]$, $t \in [0, n-1]$.
  $\texttt{away}_{w,p,t} = 1$ iff team t plays away in week w, period p.

3.1.2 Circle Method

The circle method model shares the same core variables $\texttt{home}_{w,p,t}$ and $\texttt{away}_{w,p,t}$, with additional channeling variables that link the pre-computed match pairings to period assignments:

- $\texttt{match}_{i,j,w,p} \in \{0,1\}$ where $(i,j) \in \text{matches}_w$, $w \in [0, n-2]$, $p \in [0, \lfloor n/2 \rfloor - 1]$.
  $\texttt{match}_{i,j,w,p} = 1$ iff the match between teams i and j (from week w's circle method schedule) is assigned to period p.

These variables enable bi-directional channeling between the abstract match assignments and the concrete home/away slot allocations, facilitating efficient constraint propagation.

3.1.3 Optimization Variables

For the optimization phase, swap variables control the reversal of home/away designations:

- $\texttt{swap}_{p,w} \in \{0,1\}$ where $p \in [0, \lfloor n/2 \rfloor - 1]$, $w \in [0, n-2]$.
  $\texttt{swap}_{p,w} = 1$ iff the home/away assignment for the match in period p, week w is reversed from the feasible solution.

Given a feasible schedule from the decision phase, let $\texttt{orig\_home}_{t,p,w}$ indicate whether team t was originally the home team at slot (p, w). The effective home status after applying swaps is:

$$\texttt{is\_home}_{t,p,w} = \begin{cases} \neg\texttt{swap}_{p,w} & \text{if } \texttt{orig\_home}_{t,p,w} = 1 \\ \texttt{swap}_{p,w} & \text{if } \texttt{orig\_home}_{t,p,w} = 0 \end{cases} \tag{3}$$

3.2 Constraints

All constraints are encoded in propositional logic. Cardinality constraints (`exactly_one`, `at_most_k`, `at_least_k`, `exactly_k`) are implemented using the sequential counter encoding [16], which provides efficient unit propagation during CDCL solving.

3.2.1 Canonical Method

General Constraints.

- **One home and one away team per slot:**

$$\texttt{exactly\_one}(\{\texttt{home}_{w,p,t} \mid t \in T\}) \quad \forall w \in W, p \in P \tag{4}$$

$$\texttt{exactly\_one}(\{\texttt{away}_{w,p,t} \mid t \in T\}) \quad \forall w \in W, p \in P \tag{5}$$

Symmetry Breaking Constraints.

- **Fixed team n position (SB1):** For $n \geq 8$, team n plays in period $p_{\text{target}} = w \mod |P|$ during week w:

$$\texttt{home}_{w,p_{\text{target}},n-1} \vee \texttt{away}_{w,p_{\text{target}},n-1} \quad \forall w \in \{0, \ldots, |W| - 2\} \tag{6}$$

  For $n \geq 14$, team $n - 1$ is additionally fixed in period $p_{\text{next}} = (w + 1) \mod |P|$.

- **Lexicographic order among periods (SB2):** For each week w and consecutive periods $p, p + 1$:

$$\bigwedge_{j<t} (\texttt{home}_{w,p,j} = \texttt{home}_{w,p+1,j}) \Rightarrow (\neg\texttt{home}_{w,p,t} \vee \texttt{home}_{w,p+1,t}) \tag{7}$$

Implied Constraints.

- **Total matches per team (IC1):** same as section 1.1
- **Different matches per period within week (IC2):**

$$(\texttt{home}_{w,p_1,i} \wedge \texttt{away}_{w,p_1,j}) \Rightarrow \neg(\texttt{home}_{w,p_2,i} \wedge \texttt{away}_{w,p_2,j}) \quad \forall w, p_1 < p_2, i \neq j \tag{8}$$

3.2.2 Circle Method

The circle method generates match pairings deterministically for each week using the round-robin algorithm described in Section 1.2. The SAT model assigns these fixed matches to periods within their designated week.

General Constraints.

- **Channeling between match and home/away variables:**

$$\texttt{match}_{i,j,w,p} \Rightarrow (\texttt{home}_{w,p,i} \wedge \texttt{away}_{w,p,j}) \tag{9}$$

$$\neg\texttt{match}_{i,j,w,p} \Rightarrow (\neg\texttt{home}_{w,p,i} \wedge \neg\texttt{away}_{w,p,j}) \tag{10}$$

  for all matches (i, j) in week w's circle method schedule.

- **Exactly one match per period in each week:**

$$\texttt{exactly\_one}(\{\texttt{match}_{i,j,w,p} \mid (i,j) \in \text{matches}_w\}) \quad \forall w \in W, p \in P \tag{11}$$

- **Each match assigned to exactly one period:**

$$\texttt{exactly\_one}(\{\texttt{match}_{i,j,w,p} \mid p \in P\}) \quad \forall w \in W, (i,j) \in \text{matches}_w \tag{12}$$

- **Period limit:**

$$\texttt{at\_most\_k}(\{\texttt{home}_{w,p,t}, \texttt{away}_{w,p,t} \mid w \in W\}, 2) \quad \forall t \in T, p \in P \tag{13}$$

Symmetry Breaking Constraints. The fixed team n position constraint (SB1) is applied for $n \geq 8$:

$$\texttt{home}_{w,p_{\text{target}},n-1} \vee \texttt{away}_{w,p_{\text{target}},n-1} \quad \forall w \in \{0, \ldots, |W| - 2\} \tag{14}$$

where $p_{\text{target}} = w \mod |P|$. This constraint is disabled for smaller instances ($n < 8$) as it becomes overly restrictive when combined with the circle method's structural constraints, potentially leading to unsatisfiability.

Implied Constraints. The same implied constraints (IC1, IC2) as in the canonical method can be optionally enabled. However, experimental results showed that these constraints significantly increase the problem size without proportional benefits for the circle method, as the channeling constraints already provide strong propagation.

3.2.3 Optimization Phase Constraints

Given a feasible schedule from Phase 1, the optimization phase introduces:

- **First week fixed (no swaps):** The first week maintains the original home/away assignments:

$$\neg\texttt{swap}_{p,0} \quad \forall p \in P \tag{15}$$

- **Home count bounds for target imbalance d:**

$$\texttt{at\_least\_k}(\{\texttt{is\_home}_{t,p,w} \mid (p,w) \in \text{slots}_t\}, \texttt{min\_home}) \quad \forall t \in T \tag{16}$$

$$\texttt{at\_most\_k}(\{\texttt{is\_home}_{t,p,w} \mid (p,w) \in \text{slots}_t\}, \texttt{max\_home}) \quad \forall t \in T \tag{17}$$

3.3 Validation

3.3.1 Experimental Design

The SAT model was implemented using the Z3 solver [12]. For solver-independent validation, the model was exported to standard DIMACS [13] CNF format using Z3's `tseitin-cnf` tactic, allowing validation with external SAT solvers (MiniSat, Glucose) via the PySAT library. Variable mappings between Z3 symbolic names and DIMACS integer indices were maintained for solution extraction. The solvers have been tested for different combinations of the implied and symmetry breaking constraints, to test which one perform better for our model.

3.3.2 Experimental Results

Tables 3 and 4 show the performance of Z3 with different constraint configurations for the canonical and circle methods respectively. Each cell reports decision_time|optimization_time, where N/A indicates timeout (300s).

ID	Base	SB	IC	SB2+IC	All
2	0|**0**	0|**0**	0|**0**	0|**0**	0|**0**
4	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
6	1|**1**	3|**1**	7|**6**	5|**9**	5|**5**
8	16|**15**	19|**10**	43|**27**	43|**29**	43|**31**
10	53|**47**	52|**48**	105|N/A	105|**131**	104|**132**
12	N/A	227|**190**	N/A	N/A	288|N/A
14	N/A	N/A	N/A	N/A	N/A

Table 3: Results implementing **canonical method**.

ID	Z3 (base)	Z3 + SB1	Z3 + IC2	Z3 + SB1 + IC
2	0|**0**	0|**0**	0|**0**	0|**0**
4	UNSAT	UNSAT	UNSAT	UNSAT
6	0|**0**	0|**0**	0|**1**	2|**6**
8	1|**2**	1|**1**	1|**3**	15|**18**
10	3|**4**	3|**3**	8|**8**	57|**56**
12	6|**7**	7|**5**	25|**25**	151|**145**
14	8|**16**	10|**17**	64|**60**	N/A
16	41|**33**	43|**21**	138|**157**	N/A
18	90|**87**	31|**50**	226|N/A	N/A
20	N/A	122|**195**	N/A	N/A

Table 4: Results implementing **circle method**.

	Circle Method		Canonical Method	
ID	MiniSat	Glucose	MiniSat	Glucose
6	0	0	2	2
8	3	2	N/A	N/A
10	N/A	N/A	N/A	N/A

Table 5: Results (in seconds) using DIMACS solvers (MiniSat, Glucose) via PySAT, decision version only.

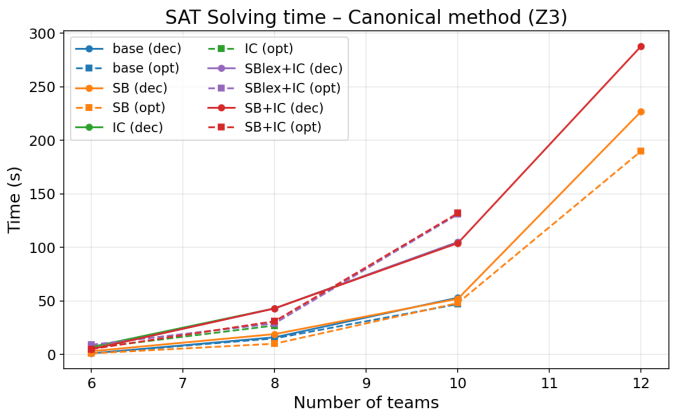

Figure 4: SAT solving time for the canonical method using Z3. Solid lines represent decision versions, dashed lines represent optimization versions.

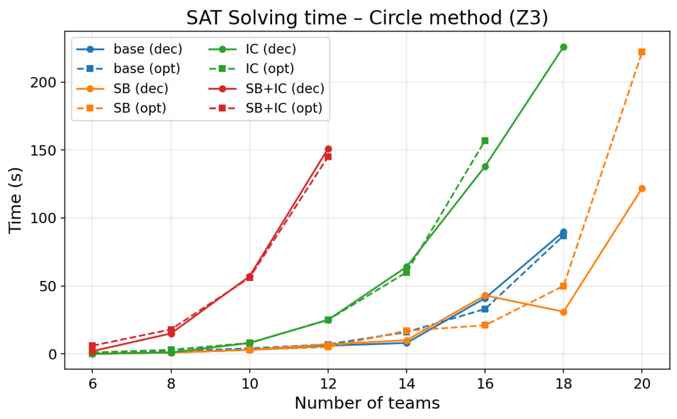

Figure 5: SAT solving time for the circle method using Z3. Solid lines represent decision versions, dashed lines represent optimization versions.

4 SMT Model

Satisfiability Modulo Theories (SMT) determines whether a logical formula is satisfiable with respect to background theories such as arithmetic, arrays, or bit-vectors [17] [18]. SMT was implemented exploiting two different solvers, Z3 [12] and CVC5 [19].

4.1 Decision Variables

4.1.1 Canonical method

- **Match variables:** $x_{w,p,h,a} \in \{0,1\}$ for $w \in [0, n-2]$, $p \in [0, \lfloor n/2 \rfloor - 1]$, $h, a \in [0, n-1]$ with $h \neq a$. True iff team h hosts team a in week w, period p.
- **Swap variables:** $\text{swap}_{p,w} \in \{0,1\}$ for $p \in [0, \lfloor n/2 \rfloor - 1]$, $w \in [0, n-2]$. True iff the match in period p, week w reverses home/away assignments.
- **Count variables:** $\text{home\_count}_t, \text{away\_count}_t \in \mathbb{Z}$ for $t \in [0, n-1]$. Number of home and away games for team t.

4.1.2 Circle Method

- **Assignment variables:** $\text{assign}_{p,w} \in [0, \lfloor n/2 \rfloor - 1]$ where $p \in [0, \lfloor n/2 \rfloor - 1]$, $w \in [0, n-2]$.
  $\text{assign}_{p,w} = k$ iff the k-th match of week w is assigned to period p.
- **Position variables:** $\text{pos}_{t,w} \in [0, \lfloor n/2 \rfloor - 1]$ where $t \in [1, n]$, $w \in [0, n-2]$.
  $\text{pos}_{t,w} = p$ iff team t's match in week w is placed in period p.
- **Count variables:** $\text{count}_{t,p} \in [0, n-1]$ where $t \in [1, n]$, $p \in [0, \lfloor n/2 \rfloor - 1]$.
  Number of times team t appears in period p across all weeks.
- **Swap variables:** $\text{swap}_{p,w} \in \{0,1\}$ where $p \in [0, \lfloor n/2 \rfloor - 1]$, $w \in [0, n-2]$.
  $\text{swap}_{p,w} = 1$ iff match in period p, week w reverses home/away assignments.

4.2 Constraints

4.2.1 Canonical method

Implied constraint

- **Balanced periods:** $\sum_{w,h\neq a} x_{w,p,h,a} = n-1$ for all p. Each period hosts $n-1$ matches across all weeks.

Symmetry breaking

- **Symmetry breaking:** $x_{0,0,0,1} = 1$. Team 0 hosts team 1 in week 0, period 0.

4.2.2 Circle Method

- **Assignment domain:** $\text{assign}_{p,w} \in [0, \lfloor n/2 \rfloor - 1]$ for all $p \in [0, \lfloor n/2 \rfloor - 1], w \in [0, n-2]$.
- **Weekly permutation:** $\bigwedge_{w=0}^{n-2} \bigwedge_{\substack{p_1,p_2 \in [0,\lfloor n/2 \rfloor -1] \\ p_1 \neq p_2}} (\text{assign}_{p_1,w} \neq \text{assign}_{p_2,w})$.
- **Channeling:** $\bigwedge_{t=1}^{n} \bigwedge_{w=0}^{n-2} \bigwedge_{p=0}^{\lfloor n/2 \rfloor -1} \left[(\text{assign}_{p,w} = m_{t,w}) \Leftrightarrow (\text{pos}_{t,w} = p)\right]$ where $m_{t,w} \in [0, \lfloor n/2 \rfloor - 1]$ is the match index containing team t in week w.
- **Period capacity:** $\bigwedge_{p=0}^{\lfloor n/2 \rfloor -1} \bigwedge_{t=1}^{n} \left(\sum_{w=0}^{n-2} [\text{pos}_{t,w} = p] \leq 2\right)$ where $[\cdot]$ denotes the Iverson bracket.

Implied constraints

- **Count definition:** $\bigwedge_{t=1}^{n} \bigwedge_{p=0}^{\lfloor n/2 \rfloor -1} \left(\text{count}_{t,p} = \sum_{w=0}^{n-2} [\text{pos}_{t,w} = p]\right)$ with $\text{count}_{t,p} \in [0, n-1]$. Auxiliary variables counting team t's appearances in period p.
- **Weekly participation:** $\bigwedge_{t=1}^{n} \left(\sum_{p=0}^{\lfloor n/2 \rfloor -1} \text{count}_{t,p} = n-1\right)$. Each team plays exactly once per week across all periods.
- **Global balance:** $\sum_{t=1}^{n} \sum_{p=0}^{\lfloor n/2 \rfloor -1} \text{count}_{t,p} = 2 \cdot \lfloor n/2 \rfloor \cdot (n-1)$. Total team appearances equals twice the number of matches (each match involves two teams).

Symmetry breaking

- **Fix period:** $\text{pos}_{n,w} = w \bmod \lfloor n/2 \rfloor$ for all w. Team n plays in specific periods.
- **Lexicographic order:** $[\text{assign}_{p,0}, \ldots, \text{assign}_{p,n-2}] \leq_{\text{lex}} [\text{assign}_{p+1,0}, \ldots, \text{assign}_{p+1,n-2}]$ for all $p < \lfloor n/2 \rfloor - 1$. Period vectors ordered lexicographically.

4.3 Validation

Figure 6 shows the solver performance for the canonical method, while figure 7 shows the performance in the circle method.

ID	Z3 w/o SB	Z3 with SB	CVC5 w/o SB	CVC5 with SB
2	0|**0**	0|**0**	0|**0**	0|**0**
4	`UNSAT`	`UNSAT`	`UNSAT`	`UNSAT`
6	0|**0**	0|**0**	2|**2**	1|**1**
8	0|**0**	0|**0**	`N/A`	`N/A`
10	16|**16**	2|**2**	`N/A`	`N/A`
12	`N/A`	242|**225**	`N/A`	`N/A`
14	`N/A`	`N/A`	`N/A`	`N/A`
16	`N/A`	`N/A`	`N/A`	`N/A`
18	`N/A`	`N/A`	`N/A`	`N/A`

Table 6: Canonical method

ID	Z3 w/o SB	Z3 with SB	CVC5 w/o SB	CVC5 with SB
2	0|**0**	0|**0**	0|**0**	0|**0**
4	`UNSAT`	`UNSAT`	`UNSAT`	`UNSAT`
6	0|**0**	0|**0**	0|**0**	0|**0**
8	0|**0**	0|**0**	0|**0**	0|**0**
10	0|**0**	0|**0**	2|**1**	0|**1**
12	0|**0**	0|**0**	16|**6**	6|**6**
14	1|**5**	1|**5**	13|**24**	11|**13**
16	13|**94**	13|**94**	178|**71**	37|**42**
18	272|**62**	57|**62**	`N/A`	37|`N/A`

Table 7: Circle method

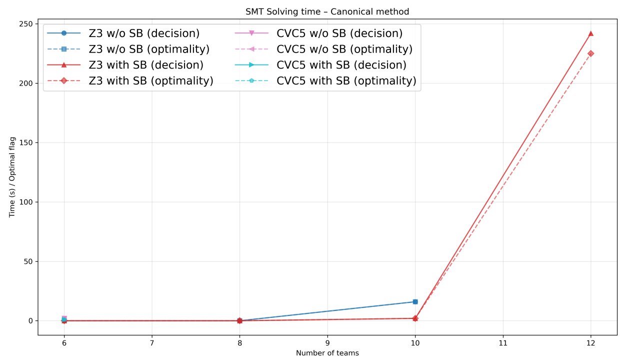

Figure 6: SMT solving time – Canonical method with different solver configurations.

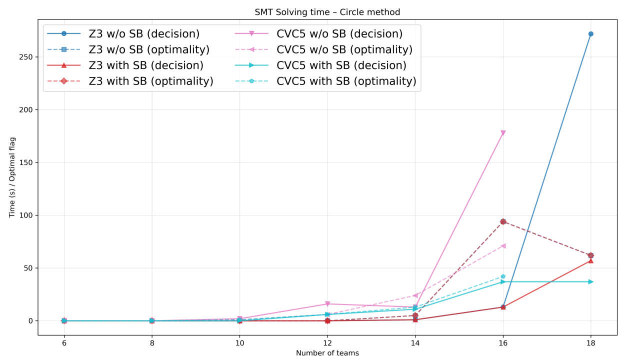

Figure 7: SMT solving time – Circle method with different solver configurations.

5 MIP Model

Mixed-Integer Programming (MIP) is a mathematical optimization framework used to model decision-making problems that involve both continuous and discrete variables. A MIP model consists of a linear objective function to be minimized or maximized, subject to a set of linear constraints. While continuous variables can take any real values within given bounds, integer variables, often restricted to binary values, are used to represent logical decisions such as on/off choices, assignments, or sequencing [20] [21].

5.1 Decision variables

5.1.1 Canonical Method

In the canonical method, MIP model is based on binary decision variables that represent the occurrence of a match in a precise slot:

$$x_{i,j,w,p} \in \{0,1\} \quad \forall i,j,w,p \quad i \neq j \tag{18}$$

where $x_{i,j,w,p} = 1$ if team i plays against team j in week w, period p, with team i playing at home and team j playing away and $x_{i,j,w,p} = 0$ otherwise.

5.1.2 Circle Method

In the circle method, since matches are predetermined for each week, the main decision variable assigns each match to a period within its fixed week:

$$x_{m,p} \in \{0,1\}, \quad \forall m \in \text{MATCHES},\ p \in P \text{ for which match } m \text{ occurs.} \tag{19}$$

Here, $x_{m,p} = 1$ if match $m = (t_1, t_2)$ takes place in period p. The week of each match is given by the parameter `match_week`.

Furthermore, for both methods, auxiliary variables were defined to linearize the objective function, as explained in section 1.1.

5.2 Constraints

The model formalization described in section 1.1 has been followed also in the MIP model. Different constraints have been used for the canonical method and circle method. All the constraints described below are linear.

5.2.1 Canonical Method

General Constraints For the canonical method, the following constraints have been used:

- **Given a time slot, at most one of the two possible home/away combinations between each pair of teams may occur**

$$x_{i,j,w,p} + x_{j,i,w,p} \leq 1 \quad \forall w,p,i,j,\ i<j \tag{20}$$

Symmetry Breaking Constraints

- **Team symmetry:** Fix the schedule of the first week to break team symmetries:

$$x_{2p-1,2p,1,p} = 1 \quad \forall p \tag{21}$$

$$x_{i,j,1,p} = 0 \quad \forall p,\ (i,j) \neq (2p-1,2p),\ i \neq j \tag{22}$$

- **Team 1 opponent ordering constraint:** It implies the matchups of team 1, by fixing that in week w, team 1 plays against team $w+1$: $\sum_{p=1}^{n/2} (x_{1,w+1,w,p} + x_{w+1,1,w,p}) = 1 \quad \forall w$

5.2.2 Circle Method

General Constraints

- **Each period has exactly one match:** $\sum_{p\in P} x_{m,p} = 1, \quad \forall m \in \text{MATCHES}$
- **Every team plays at most twice in the same period over the tournament:**

$$\sum_{m=(t_1,t_2)\in \text{MATCHES}} x_{m,p} \leq 2, \quad \forall t \in T,\ p \in P \tag{23}$$

Symmetry Breaking Constraints

- **The periods of matches in first week are assigned in the order obtained by the circle method**: $x_{m,p_m} = 1 \quad \forall m \in \text{MATCHES with week}(m) = 1$

Implied Constraints

- **Each period in each week has exactly one match:** $\sum_{\substack{m \in \text{MATCHES} \\ \text{week}(m)=w}} x_{m,p} = 1 \quad \forall w \in W,\ p \in P$

5.3 Validation

The MIP model was first built using AMPL [4], an algebraic modeling language for building and deploying optimization applications. AMPL was chosen in order to exploit a solver-independent language which can be tested by multiple solvers without modifying the program design. In particular, four different solvers were used:

- CBC [22]
- HiGHS [23]
- CPLEX [24]
- GUROBI [25]

The solvers were applied to the model in Python, thanks to the package amplpy, which allows to work with .mod AMPL files in Python.

Solvers were tested exploiting the two different schedule formulation techniques from section 1.2. For both the canonical method and circle method, all the different combinations of the symmetry breaking constraints have been tested, as from sections 5.2.1 and 5.2.2. As shown in table 8 and table 9, implementing the circle method has led to significantly better results, leading to obtain a solution until $n = 22$, exploiting the HiGHS solver. Moreover, with the circle method, CBC solver has shown worse performance with respect to other solvers similar performance.

ID	HiGHS w/o SB	HiGHS with SB	CBC w/o SB	CBC with SB	CPLEX w/o SB	CPLEX with SB	GUROBI w/o SB	GUROBI with SB
2	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
4	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
6	0 | **0**	0 | **0**	0 | N/A	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
8	0 | **6**	0 | **3**	0 | N/A	0 | N/A	0 | **0**	0 | **0**	0 | **0**	0 | **0**
10	7 | **15**	4 | **16**	2 | N/A	0 | N/A	1 | **2**	0 | **1**	0 | **4**	0 | **3**
12	174 | N/A	0 | **97**	20 | N/A	2 | N/A	8 | **18**	1 | **9**	4 | **232**	1 | **7**
14	N/A | N/A	N/A | N/A	13 | N/A	N/A | N/A	261 | **72**	54 | N/A	16 | N/A	7 | N/A
16	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	108 | N/A

Table 8: Results implementing the **canonical method**.

ID	HiGHS w/o SB	HiGHS with SB	CBC w/o SB	CBC with SB	CPLEX w/o SB	CPLEX with SB	GUROBI w/o SB	GUROBI with SB
2	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
4	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT	UNSAT
6	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
8	0 | **0**	0 | **0**	0 | N/A	0 | **0**	0 | **0**	0 | **0**	0 | **1**	0 | **0**
10	0 | **1**	0 | **0**	1 | **1**	0 | **1**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
12	0 | **1**	1 | **1**	0 | **4**	1 | **0**	0 | **0**	0 | **0**	0 | **0**	0 | **0**
14	0 | **16**	0 | **6**	0 | **67**	2 | **7**	0 | **0**	0 | **0**	3 | **0**	5 | **6**
16	29 | **4**	119 | **97**	138 | N/A	N/A | N/A	3 | **3**	12 | **6**	15 | **14**	72 | **79**
18	72 | **8**	183 | **269**	N/A | N/A	N/A | N/A	4 | **4**	78 | **84**	N/A | **56**	N/A | **69**
20	173 | **84**	N/A | **142**	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A
22	261 | **224**	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A	N/A | N/A

Table 9: Results implementing the **circle method**.

Figure 8 shows the behaviour of each solver in the decision and optimization versions when exploiting the canonical method. Similarly, figure 9 shows the performance when exploiting the circle method.

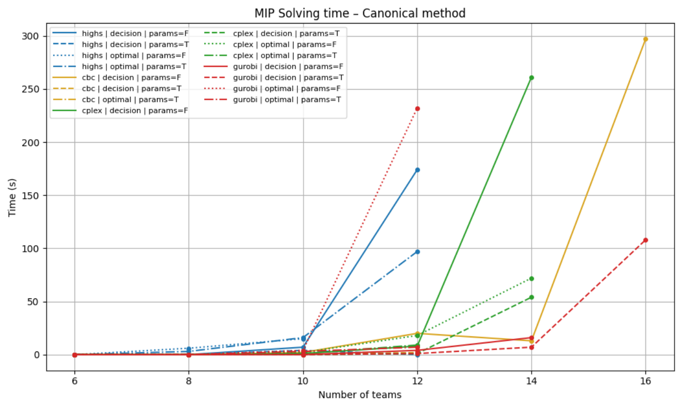

Figure 8: Solving time for different constraint configurations with MIP solvers using canonical method.

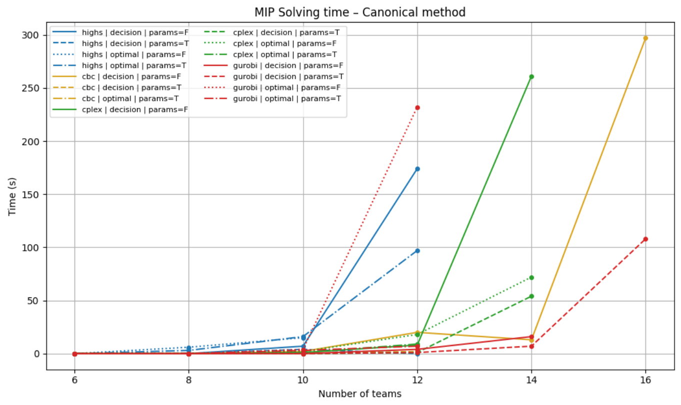

Figure 9: Solving time for different constraint configurations with MIP solvers using circle method.

6 Conclusions

This comparative study reveals several important insights about multi-paradigm optimization for the Sports Tournament Scheduling problem:

- Circle method superiority: The circle method consistently outperforms the canonical approach across all paradigms, enabling solutions for significantly larger instances
- Scale-dependent symmetry breaking: Symmetry breaking constraints show diminishing or negative returns on small instances but become increasingly beneficial as problem size grows, particularly evident in SAT where the SB1 constraint reduces solving time from 90s to 31s at n=18.
- Solver-specific optimizations matter: Z3's native cardinality propagators significantly outperform external DIMACS solvers, and commercial MIP solvers (CPLEX, GUROBI) show superior performance compared to open-source alternatives on medium-sized instances.

7 Authenticity and Author Contribution Statement

We declare that the work described in this report is our own and has not been copied from any other source. All external ideas, resources, or materials that have been used have been appropriately cited and referenced in the bibliography.

AI-generated content has been used in this report and it was limited to language and formulae refinement, conceptual clarification, without compromising academic integrity.

7.1 Author Contributions

All authors contributed to every model development, exploiting GitHub branches in the remote repository. In particular:

- **Matteo Preda** contributed mainly to the **CP model**
- **Marco Zocco Ramazzo** contributed mainly to the **SAT model**
- **Raffaele Sali** contributed mainly to the **MIP model**

References

[1] Celso C Ribeiro. Sports scheduling: Problems and applications. *International Transactions in Operational Research*, 19(1-2):201–226, 2012.

[2] Thomas P Kirkman. On a problem in combinations. *Cambridge and Dublin Mathematical Journal*, 2:191–204, 1847.

[3] R. V. Rasmussen and M. A. Trick. Round robin scheduling – a survey. *European Journal of Operational Research*, 188(3):617–636, 2008. Available as PDF at m̈at.tepper.cmu.edu/trick/survey.pdf·.

[4] Robert Fourer, David M Gay, and Brian W Kernighan. Ampl: A mathematical programming language. *Management Science*, 36(5):519–554, 1990.

[5] E. Lambrechts, A. M. C. Ficker, D. R. Goossens, and F. C. R. Spieksma. Round-robin tournaments generated by the circle method have maximum carry-over. *Mathematical Programming*, 172(1–2):277–302, 2018.

[6] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[7] Krzysztof Apt. *Principles of constraint programming*. Cambridge university press, 2003.

[8] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling language. In *International conference on principles and practice of constraint programming*, pages 529–543. Springer, 2007.

[9] Christian Schulte, Guido Tack, and Mikael Z Lagerkvist. Modeling and programming with gecode. *Schulte, Christian and Tack, Guido and Lagerkvist, Mikael*, 1, 2010.

[10] G. Chu, A. Schutt, P. J. Stuckey, and M. Wallace. Chuffed, a lazy clause generation solver. In *Principles and Practice of Constraint Programming*, volume 8656 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2014.

[11] Joao P Marques-Silva and Karem A Sakallah. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 2002.

[12] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[13] David S Johnson and Michael A Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc., 1996.

[14] Niklas Sorensson and Niklas Een. Minisat v1. 13-a sat solver with conflict-clause minimization. *SAT*, 2005(53):1–2, 2005.

[15] Gilles Audemard and Laurent Simon. On the glucose sat solver. *International Journal on Artificial Intelligence Tools*, 27(01):1840001, 2018.

[16] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *International conference on principles and practice of constraint programming*, pages 827–831. Springer, 2005.

[17] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018.

[18] Leonardo De Moura and Nikolaj Bjørner. Satisfiability modulo theories: introduction and applications. *Communications of the ACM*, 54(9):69–77, 2011.

[19] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, et al. cvc5: A versatile and industrial-strength smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442. Springer, 2022.

[20] Laurence A Wolsey. Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–10, 2007.

[21] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization: Festschrift for martin grötschel*, pages 449–481. Springer, 2013.

[22] John Forrest and Robin Lougee-Heimer. Cbc user guide. In *Emerging theory, methods, and applications*, pages 257–277. INFORMS, 2005.

[23] Qi Huangfu and JA Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018.

[24] CPLEX User's Manual. Ibm ilog cplex optimization studio. *Version*, 12(1987-2018):1, 1987.

[25] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, Houston, TX, 2024. Available at https://www.gurobi.com.