ITC2021 – Sports Timetabling
Problem Description and File Format

David Van Bulck[1], Dries Goossens[1], Jeroen Beliën[2], Morteza Davari[3]

1. Time-constrained double round-robin tournaments

The aim of this competition is to stimulate the development of solvers for the construction of round-robin timetables, meaning that each team plays against every other team a fixed number of times. Besides the round-robin format, many other timetabling formats exist. One notorious example is the knock-out format where teams are tied in pairs and the loser of each game is eliminated. Nevertheless, round-robin tournaments are probably the most researched format (see Knust [4]), and are very common in practice (see e.g. Goossens and Spieksma [3]). Most sports competitions organize a double round-robin tournament (2RR) where teams meet twice but single, triple, and even quadruple round-robin tournaments also occur. According to Nemhauser and Trick [5], there are two types of round-robin tournaments: time-constrained timetables and time-relaxed timetables. A timetable is time-constrained (also called compact) if it uses the minimal number of time slots needed, and is time-relaxed otherwise. In this competition, we only consider time-constrained double round-robin tournaments with an even number of teams. Under this setting, the total number of time slots is exactly equal to the total number of games per team, and hence each team plays exactly one game per time slot. For an example of a time-constrained 2RR timetable, we refer to Table 1.

The constraints that appear in real-life problem instances are extremely diverse: apart from some basic constraints, each competition has its own requirements. In this competition, we assume that there are two types of constraints: hard constraints represent fundamental properties of the timetable that can never be violated, while soft constraints represent preferences that should be satisfied whenever possible. While many possible optimization objectives appear in the literature, this competition considers problem instances only where the objective is to minimize the penalties from violated soft constraints. This assumption makes the problem formulation more attractive for a wider timetabling community, while retaining the empirical complexity of the problems.

The problem instances will be expressed using the standardized RobinX XML data format developed by Van Bulck et al. [7]. The main intention of this data format is to promote problem instance data sharing and reuse among different users and software applications, and this is exactly what the timetabling competition envisions. The XML data format of RobinX is open, human readable (i.e. no binary format), software and platform independent, and flexible enough

Email addresses: david.vanbulck@ugent.be (David Van Bulck), dries.goossens@ugent.be (Dries Goossens), jeroen.belien@kuleuven.be (Jeroen Beliën), morteza.davari@kuleuven.be (Morteza Davari)

[1]Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium

[2]Faculty of Economics and Business, KU Leuven, Leuven, Belgium

[3]SKEMA Business School, Lilles, France

Table 1: A time-constrained double round-robin timetable for a single league with 6 teams. Each game is represented by an ordered pair in which the first element is the home team, and the second element is the away team.

s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
(1,2)	(2,5)	(2,4)	(2,3)	(6,2)	(4,2)	(5,2)	(2,1)	(3,2)	(2,6)
(3,4)	(4,1)	(1,6)	(5,1)	(4,5)	(6,1)	(1,4)	(4,3)	(1,5)	(5,4)
(5,6)	(6,3)	(5,3)	(6,4)	(1,3)	(3,5)	(3,6)	(6,5)	(4,6)	(3,1)

```
<Instance>
  <MetaData> ... </MetaData>
  <Resources> ... </Resources>
  <Structure> ... </Structure>
  <Constraints>
        <CapacityConstraints/>
        <GameConstraints/>
        <BreakConstraints/>
        <FairnessConstraints/>
        <SeparationConstraints/>
  </Constraints>
  <ObjectiveFunction> ... </ObjectiveFunction>
</Instance>
```

Figure 1: Main structure of the problem instance XML file format.

to store the problem instances. Most of the sports timetabling constraints are easy to express in words but are hard to enforce within specific algorithms such as mathematical programming or metaheuristics. We believe this format minimizes the specification burden and maximizes the accessibility. The main advantage of XML over plain text-only file formats lies in the structured way of data storage. Indeed, an important motivation behind XML is to separate data representation from data content.

2. Problem description and problem instance file format

In this section, we outline the structure of the problem instances as they appear in the XML format[4]. The main structure of the XML files is provided in Figure 1. The remainder of this section describes each of the XML tags in more detail.

2.1. Meta data

The meta data of a problem instance (see Figure 2) consists of a unique instance name, the data type which is always artificial (A), the contributor which is always ITC2021, and the date which is always 2020.

[4]For a general introduction to XML, we refer to `https://www.w3schools.com/xml/`. Generating XML files in C++ is simple and convenient with tinyXML.

```
<MetaData>
      <InstanceName>Test Instance 1</InstanceName>
      <DataType>A</DataType>
      <Contributor>ITC2021</Contributor>
      <Date year="2020"/>
</MetaData>
```

Figure 2: XML specification for the meta data of the problem instance.

2.2. Resources

The resources in a sports timetabling problem are the teams, time slots, and a league in which all teams play (see Figure 3). We only consider problem instances with one league in which all teams play. Time slots, in time-constrained scheduling often called rounds, represent periods in time like half days or days in the season. A team can never play more than one game per time slot. The numbering of unique resource id's starts from zero on. The XML format of RobinX offers the possibility to group arbitrary sets of teams and time slots into respectively team and time groups; for simplicity, we will not make use of this functionality in the XML files of the competition. The total number of teams will be 16, 18, or 20. Since the competition is time-constrained, the total number of time slots is thus either 30, 34, or 38.

```
<Resources>
      <Leagues>
            <league id="0" name="League 0"/>
      </Leagues>
      <Teams>
            <team id="0" league="0" name="Team 1"/>
      </Teams>
      <Slots>
            <slot id="0" name="Slot 0"/>
      </Slots>
</Resources>
```

Figure 3: XML specification for the resources of the problem instance.

2.3. Structure

The structure of the competition format is always a time-constrained 2RR where each team meets every other team once at home and once away (see Figure 4). The season of a 2RR is often split into two equally long intervals that each contain a 1RR and in which the home-away status of mutual games for each pair of teams alternates between consecutive intervals (e.g. time slots s_1 to s_5 and s_6 to s_{10} in Table 1). We call a timetable that follows this format 'phased' and denote this requirement via the symbol 'P' in the gameMode tag. In case the tournament need not be phased, the gameMode tag has the value 'NULL' in the ITC2021 competition instances .

2.4. Constraints

Sports timetables need to satisfy a usually large set of constraints C, which is partitioned into hard constraints C_{hard} and soft constraints C_{soft}. For each constraint tag, this is denoted by

```
<Structure>
    <Format leagueIds="0">
        <numberRoundRobin>2</numberRoundRobin>
        <compactness>C</compactness>
        <gameMode>P</gameMode>
    </Format>
</Structure>
```

Figure 4: XML specification for the round-robin structure of the problem instance.

the type attribute which can take the values 'HARD' and 'SOFT'. Hard constraints represent fundamental properties of the timetable that can never be violated. Soft constraints, in contrast, rather represent preferences that should be satisfied whenever possible. The validation of each constraint $c \in C$ results in a vector of n_c integral numbers, called the deviation vector $D_c = [d_1\ d_2\ \ldots\ d_{n_c}]$. If a constraint is satisfied, all elements of its deviation vector are equal to zero. Contrarily, the deviation vector of a violated constraint contains one or more non-zero elements.

The instances will feature a variety of constraints from the classification format developed by Van Bulck et al. [7]. Below follows an overview of the 9 (simplified) constraints from the classification framework that we consider; other constraints than these will not be included in the ITC2021 instances. The selection of these constraints is based on their popularity in real-life competitions. For examples of the use cases of each constraint and for a more formal description, we refer to Van Bulck et al. [7].

2.4.1. Capacity constraints

Capacity constraints (CA) force a team to play home or away and regulate the total number of games played by a team or group of teams. We consider four different capacity constraints.

`CA1` <CA1 teams="0" max="0" mode="H" slots="0" type="HARD"/>

Each team from `teams` plays at most `max` home games (`mode` = "H") or away games (`mode` = "A") during time slots in `slots`. Team 0 cannot play at home on time slot 0.

Each team in `teams` triggers a deviation equal to the number of home games (`mode` = "H") or away games (`mode` = "A") in `slots` more than `max`.

Constraint CA1 is of fundamental use in sports timetabling to model 'place constraints' that forbid a team to play a home game or away game in a given time slot. Constraint CA1 can also help to balance the home-away status of games over time and teams. For example, when the home team receives ticket revenues, teams often request to have a limit on the number of away games they play during the most lucrative time slots. Note that a CA1 constraint where the set `teams` contains more than one team can be split into several CA1 constraints where the set `teams` contains one team: in all ITC2021 instances, `teams` therefore contains only one team.

`CA2` <CA2 teams1="0" min="0" max="1" mode1="HA" mode2="GLOBAL" teams2="1;2" slots="0;1;2" type="SOFT"/>

Each team in `teams1` plays at most `max` home games (`mode1` = "H"), away games (`mode1` = "A"), or games (`mode1` = "HA") against teams (`mode2` = "GLOBAL"; the only mode we consider) in `teams2` during time slots in `slots`. Team 0 plays at most one game against

teams 1 and 2 during the first three time slots.

Each team in `teams1` triggers a deviation equal to the number of home games (`mode1` = "H"), away games (`mode1` = "A"), or games (`mode1` = "HA") against teams in `teams2` during time slots in `slots` more than `max`.

Constraint CA2 generalizes CA1 and can model 'top team and bottom team constraints' that prohibit bottom teams from playing all initial games against top teams. Note that a CA2 constraint where the set `teams` contains more than one team can be split into several CA2 constraints where the set `teams` contains one team: in all ITC2021 instances, `teams` therefore contains only one team.

`CA3` <CA3 teams1="0" max="2" mode1="HA" teams2="1;2;3" intp="3" mode2= "SLOTS" type="SOFT"/>

Each team in `teams1` plays at most `max` home games (`mode1` = "H"), away games (`mode1` = "A"), or games (`mode1` = "HA") against teams in `teams2` in each sequence of `intp` time slots (`mode2` = "SLOTS"; the only mode we consider). Team 0 plays at most two consecutive games against teams 1, 2, and 3.

Each team in `teams1` triggers a deviation equal to the sum of the number of home games (`mode1` = "H"), away games (`mode1` = "A"), or games (`mode1` = "HA") against teams in `teams2` more than `max` for each sequence of `intp` time slots.

In the ITC2021 competition, there are at most two CA3 hard constraints per problem instance which limit the maximal length of home stands (and/or away trips) by forbidding consecutive home breaks (and/or consecutive away breaks). Furthermore, there can be arbitrary many soft constraints that limit the total number of consecutive games against certain strength groups of teams.

`CA4` <CA4 teams1="0;1" max="3" mode1="H" teams2="2,3" mode2="GLOBAL" slots ="0;1" type="HARD"/>

Teams in `teams1` play at most `max` home games (`mode1` = "H"), away games (`mode1` = "A"), or games (`mode1` = "HA") against teams in `teams2` during time slots in `slots` (`mode2` = "GLOBAL") or during each time slot in `slots` (`mode2` = "EVERY"). Teams 0 and 1 together play at most three home games against teams 2 and 3 during the first two time slots.

The set `slots` (`mode2` = "GLOBAL") or each time slot in `slots` (`mode2` = "EVERY") triggers a deviation equal to the number of games (i, j) (`mode1` = "H"), (j, i) (`mode1` = "A"), or (i, j) and (j, i) (`mode1` = "HA") with i a team from `teams1` and j a team from `teams2` more than `max`.

In contrast to CA2 and CA3 that define restrictions for each team in `teams1`, CA4 considers `teams1` as a single entity. This constraint is typically used to limit the total number of games between top teams, or to limit the total number of home games per time slot when e.g. two teams share a stadium.

2.4.2. Game constraints

Game constraints enforce or forbid specific assignments of a game to time slots.

GA1 <GA1 min="0" max="0" meetings="0,1;1,2;" slots="3" type="HARD"/>

At least `min` and at most `max` games from $G = \{(i_1, j_1), (i_2, j_2), \dots\}$ take place during time slots in `slots`. Game $(0, 1)$ and $(1, 2)$ cannot take place during time slot 3.

The set `slots` triggers a deviation equal to the number of games in `meetings` less than `min` or more than `max`.

Constraint GA1 deals with fixed and forbidden game to time slot assignments. Examples include the police that forbid to play high risk games during time slots in which other major events are planned, and broadcasters that request at least one 'top game' or 'classic game' in each televised time slot.

2.4.3. Break constraints

If a team plays a game with the same home-away status as its previous game, we say it has a break. As an example, team 2 in Table 1 has a home break in time slot s_3 and s_4. Breaks usually are undesired since they have an adverse impact on game attendance (see [2]) and they can be perceived as unfair due to the home advantage (e.g., [6]). Break constraints therefore regulate the frequency and timing of breaks in a competition.

BR1 <BR1 teams="0" intp="0" mode2="HA" slots="1" type="HARD"/>

Each team in `teams` has at most `intp` home breaks (`mode2` = "H"), away breaks (`mode2` = "A"), or breaks (`mode2` = "HA") during time slots in `slots`. Team 0 cannot have a break on time slot 1.

Each team in `teams` triggers a deviation equal to the difference in the sum of home breaks, away breaks, or breaks during time slots in `slots` more than `max`.

The BR1 constraint can forbid breaks at the beginning or end of the season, or can limit the total number of breaks per team. Note that a BR1 constraint where the set `teams` contains more than one team can be split into several BR1 constraints where the set `teams` contains one team: in all ITC2021 instances, `teams` therefore contains only one team.

BR2 <BR2 homeMode="HA" teams="0;1" mode2="LEQ" intp="2" slots="0;1;2;3" type="HARD "/>

The sum over all breaks (`homeMode` = "HA", the only mode we consider) in `teams` is no more than (`mode2` = "LEQ", the only mode we consider) `intp` during time slots in `slots`. Team 0 and 1 together do not have more than two breaks during the first four time slots.

The set `teams` triggers a deviation equal to the number of breaks in the set `slots` more than `intp` .

Constraint BR2 can be used to limit the total number of breaks for a subset of teams. In real-life, this constraint is mostly used to limit the total number of breaks in the competition.

2.4.4. Fairness constraints

To increase the fairness and attractiveness of competitions, the following constraint can be used.

`FA2` <FA2 teams="0;1;2" mode="H" intp="1" slots="0;1;2;3" type="HARD"/>

Each pair of teams in `teams` has a difference in played home games (`mode` = "H", the only mode we consider) that is not larger than `intp` after each time slot in `slots`. The difference in home games played between the first three teams is not larger than 1 during the first four time slots.

Each pair of teams in `teams` triggers a deviation equal to the largest difference in played home games more than `intp` over all time slots in `slots`.

Constraints FA2 is typically used to enforce that a timetable is `intp`-ranking-balanced, meaning that the difference in played home games between any two teams is smaller than `intp` at any point in time.

2.4.5. Separation constraints

Separation constraints regulate the number of rounds between consecutive games involving the same teams.

`SE1` <SE1 teams="0;1" min="5" mode1="SLOTS" type="HARD"/>

Each pair of teams in `teams` has at least `min` time slots (`mode1` = "SLOTS", the only mode we consider) between two consecutive mutual games. There are at least 5 time slots between the mutual games of team 0 and 1.

Each pair of teams in `teams` triggers a deviation equal to the sum of the number of time slots less than `min` or more than `max` for all consecutive mutual games.

SE1 is typically used to express that two games with the same opponents are separated by at least a given number of time slots. If a separation of at least one time slot is required, this constraint is often called the 'no-repeater' constraint (see [1]).

When teams 0 and 1 play against each other on time slots s_1 and s_2 and there should be k time slots in between, the formula to deviate the deviation is $\max(k - (s_2 - s_1 - 1); 0)$ (and NOT $\max(k - (s_2 - s_1); 0)$ as claimed in the RobinX paper).

2.5. Objective function

Each constraint $c \in C$ triggers a penalty $p_c = w_c \sum_{i=1}^{n_c} d_i$ that is equal to the sum of the elements of the deviation vector multiplied with weight w_c (denoted by the attribute `penalty` in the soft constraint tags). The objective we use for the ITC2021 problem instances sums over all violated soft constraint penalties, that is $\sum_{c \in C_{\text{soft}}} p_c$.

3. Solution file format

A second XML standard is used to store solutions to problem instances (see Figure 5). The meta data tag stores the name of the instance XML file, the name of the generated solution, and the objective value which consists of the sum of violated hard constraints penalties (infeasibility attribute, which should be zero) and the sum of violated soft constraints penalties (objective attribute). Next comes a games tag which enumerates the time slot assigned to each game of the tournament.

```
<Solution>
  <MetaData>
    <InstanceName>Utopia.xml</InstanceName>
    <SolutionName>UtupiaSol_TeamX</SolutionName>
    <ObjectiveValue infeasibility="0" objective="2"/>
  </MetaData>
  <Games>
    <ScheduledMatch home="1" away="2" slot="1">
    ...
  </Games>
</Solution>
```

Figure 5: XML specification for a solution to a problem instance.

References

[1] K. Easton, G. Nemhauser, and M. Trick. The traveling tournament problem description and benchmarks. In T. Walsh, editor, *Principles and Practice of Constraint Programming — CP 2001*, pages 580–584, Berlin, Heidelberg, 2001. Springer.

[2] D. Forrest and R. Simmons. New issues in attendance demand: The case of the English football league. *J. Sports Econ.*, 7:247–266, 2006.

[3] D. R. Goossens and F. C.R. Spieksma. Soccer schedules in Europe: an overview. *J. Sched.*, 15:641–651, 2011.

[4] S. Knust. Classification of literature on sports scheduling. `http://www2.inf.uos.de/knust/sportssched/sportlit_class/`, 2020. Accessed: 25/03/2020.

[5] G. L. Nemhauser and M. A. Trick. Scheduling a major college basketball conference. *Oper. Res.*, 46:1–8, 1998.

[6] R. Pollard and G. Pollard. Long-term trends in home advantage in professional team sports in North America and England (1876–2003). *J. Sport. Sci.*, 23(4):337–350, 2005.

[7] D. Van Bulck, D. Goossens, J. Schönberger, and M. Guajardo. RobinX: A three-field classification and unified data format for round-robin sports timetabling. *Eur. J. Oper. Res.*, 280:568 – 580, 2020.