A Hybrid Model to Find Schedules for Double Round Robin Tournaments With Side Constraints

Jasper van Doornmalen · Christopher Hojny · Roel Lambers · Frits Spieksma

Abstract We describe a method for finding solutions to the instances provided by the International Timetabling Competition, edition 2021.

Keywords Sport scheduling · Mixed Integer Programming · Matheuristics · Fix and Optimise · International Timetabling Competition 2021

1 Introduction

The International Timetabling Competition (ITC) is a competition where participants are asked to solve instances of a particular type of timetabling problems. In the 2021 edition [4], the instances originate from the domain of sport scheduling. The aim of this note is to describe the method that we have used to find solutions for the instances provided by the 2021 edition of the ITC.

There are many types of scheduling problems within the realm of sport scheduling, however the ITC focused solely on the scheduling of so-called *Double Round Robin (DRR)* tournaments. In a DRR tournament each team plays each other team twice, once at home and once away. The matches are distributed over different *rounds* or *slots*, such that each team plays at most one match per round. When the number of teams (N) is even, it is possible to schedule a DRR using $2N - 2$ rounds, implying that each team plays exactly once in each round. The resulting schedule is then called *compact*, as it uses the minimum number of rounds needed to schedule all the matches. A compact DRR is a very popular format used in many sports; for instance, most national soccer leagues are organised as a compact DRR. As the instances provided by

Eindhoven University of Technology
Combinatorial Optimization Group
PO Box 513
5600 MB Eindhoven, The Netherlands
E-mail: {m.j.v.doornmalen, c.hojny, r.lambers, f.c.r.spieksma}@tue.nl

the ITC ask for compact DRRs, we focus here exclusively on finding compact schedules.

Literature Review

There is a growing amount of papers dealing with all kinds of problems in sport scheduling. Results on the complexity of finding round robin schedules can be traced back to Easton [5], see also Briskorn et al. [3], and Van Bulck and Goossens [12]. Integer programming formulations have been studied, among others, in Briskorn and Drexl [2]. For more general information concerning sport scheduling, we refer to well-known surveys by Rasmussen and Trick [11] and Kendall et al. [8]; a bibliography is maintained by Knust [9]. When scheduling a league in practice, many different real-life aspects may turn up, see Goossens and Spieksma [6] for an overview devoted to European soccer leagues. The instances provided by the ITC feature many practical side-constraints; in fact, we discuss the properties of a schedule as required by the 2021 ITC in Section 2, and we call the problem of finding such a schedule ITC-DRR. In Section 3 we describe our method, and in Section 4 we give the results.

2 Requirements of a ITC-DRR Schedule

Ultimately, the quality of a schedule depends on the preferences of the organisers. It is a fact, however, that there are reoccurring themes that often need to be taken into account when devising a high-quality compact DRR schedule. In Section 2.1, we specify our notation and phrase the problem, and in Section 2.2 we discuss the type of hard and soft constraints present in the instances of the ITC.

2.1 Notation

We use $\mathcal{T}$ for the set of teams, $\mathcal{S}$ for the set of rounds; thus we have $|\mathcal{T}| = N$ and $|\mathcal{S}| = 2N - 2$. A match is an ordered pair $(i, j) \in \mathcal{T} \times \mathcal{T}, i \neq j$, where team i plays Home and j plays Away. For each match, there should be a round $r \in \mathcal{S}$ where this specific match occurs and in each $r \in \mathcal{S}$, team $i \in \mathcal{T}$ should play exactly one match.

All other constraints that occur, tend to be specific for a team or a subset of teams, and applicable to a subset of the rounds, see Section 2.2. Constraints are either so called *hard constraints*, or are *soft constraints*. The hard constraints need to be satisfied; for the soft constraints, a penalty for every (unit of) violation is given. The objective is to find a schedule that satisfies all the hard constraints and minimises the total sum of the penalties induced by the soft constraints; we refer to this problem as ITC-DRR.

2.2 On the variety of constraints in the ITC instances

We describe different types of constraints that are present in the ITC instances.

2.2.1 The type of DRR

It is quite natural to organise the DRR such that each pair of teams meets once in the first half of the schedule (i.e., in the first $N-1$ rounds), and once in the second half of the schedule (i.e., in the last $N-1$ rounds); this is called *phased*. A phased competition can be regarded as two consecutive Single Round Robins (SRR).

Apart from being phased or not, most competitions require that the rounds where the match (i,j) and its return (j,i) for $i,j \in \mathcal{T}$ is played, are separated by a given minimum number of rounds. When this minimum number of rounds equals $N-1$ for all the matches, the second half of the season is the complement of the first half of the season, and the resulting DRR is called *mirrored*.

2.2.2 Constraints concerning Home Away Patterns (HAPs)

Consider, for a given team, the series of $2N-2$ home and away matches: we call the resulting pattern the Home Away Pattern (or HAP) of that team. When a team plays two matches in consecutive rounds either both home or both away, this is called a *break*. In many applications, breaks are seen as unfavourable for the team and their occurrence should be minimised.

More generally, it is quite common to demand that the home-away pattern according to which a team plays is balanced for certain sets of rounds. For instance, it is very common to demand that each team starts its first two rounds with a home match and an away match, and also plays in its last two (and even four) matches once at home and once away (twice at home and twice away). In addition, even when the DRR is not phased, one can demand that after $N-1$ rounds the number of home matches played by each team is either $\lfloor \frac{N-1}{2} \rfloor$ or $\lceil \frac{N-1}{2} \rceil$.

To ensure favourable break-patterns, schedulers often use a so-called First-Break-Then-Schedule approach, which goes back to Nemhauser and Trick [10]. In such an approach, first the home away patterns are fixed, and then the matches are determined consistent with the given HAPs.

2.2.3 Constraints concerning matches

In practice, not all matches are treated equally: matches between top teams are subject to much more attention than other matches. As a consequence, finding the appropriate round for such a match can be an important factor in the quality of a schedule. For instance, many leagues feature so-called Super Sundays: rounds where 4 or 6 top teams play matches amongst them. Finding the ideal round for such a Super Sunday is important. In addition, some matches should not be played in certain rounds: a match between top teams

should not be played in the last round (for risk of not being relevant anymore). Further, one can imagine, in the context of soccer leagues, that qualification for other tournaments have an impact on the rounds where particular matches should be avoided.

It is also very common to strive for a balance in the strength of consecutive opponents, especially in the first 4 or 6 rounds. Indeed, it is considered quite unfavourable to have to play against 3 top teams in a row, whether or not at home or away.

3 Solution process

In principle, the ITC-DRR problem can be modelled as a compact mixed-integer program (MIP), which can be solved by a black-box MIP solver. In practice, however, state-of-the-art solvers fail to solve instances even with a small number of teams within weeks. For this reason, we suggest a heuristic approach to find solutions satisfying all hard constraints of ITC-DRR and that aims to minimise the violation of soft constraints. Our approach consists of three phases:

1. constructing a schedule S for a compact phased double-round robin tournament;
2. turning S into a schedule S' adhering to all hard constraints;
3. starting from S', searching for a schedule that violates less soft constraints while still satisfying all hard constraints.

We use an explicit rule to construct the initial schedule S in Phase 1, whereas Phases 2 and 3 make use of procedure that alternates between solving a MIP and using a local search algorithm to find better solutions. In the following, we describe the details of these three steps.

3.1 Finding an Initial Schedule

To construct the initial schedule S, we use the circle method [1] to create a schedule for a compact single-round robin tournament. This tournament is then duplicated where the home-away status of each match is flipped. By combining these two single-round robin tournaments a compact and phased double-round robin tournament is found.

Note that S is only guaranteed to satisfy the compactness constraint of ITC-DRR. If S violates some hard constraint, we enter Phase 2; otherwise, we directly continue with Phase 3.

3.2 Finding Improving Schedules

In both Phase 2 and 3, our aim is to find schedules that improve on the initial schedule S or S', respectively. To quantify the quality of a solution,

we introduce a function f that measures the violation of hard constraints (Phase 2) or soft constraints (Phase 3) by a schedule. Moreover, violations of hard constraints are not permitted in Phase 3. Our goal is to find a schedule that minimises the value of f. As mentioned above, we search for such a schedule by an alternating procedure whose components are described next.

3.2.1 Local Search

Januario et al. [7] discuss how schedules for compact single-round robin tournaments can be encoded in an edge-coloured complete undirected graph. The vertices of the complete graph correspond to the teams and edges encode possible matches. Rounds of the matches are distinguished by assigning each edge a colour, i.e. the edges of each colour class form a perfect matching.

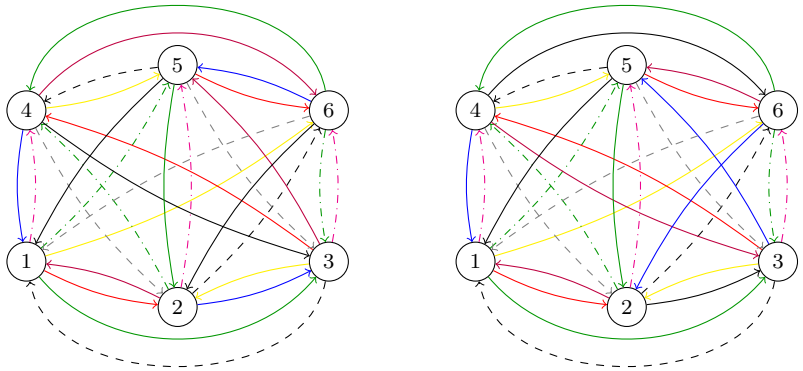
To find better schedules, they discuss a local search algorithm on such colourings. The idea of this algorithm is that one schedule can be turned into another by the following procedure: Starting with two vertices (teams) $i, j \in \mathcal{T}$ one can select a set of colours (rounds) $c_0, c_1, \ldots, c_{p-1} \in \mathcal{S}$ that induce p disjoint paths $C_0, \ldots, C_{p-1}$ where path C_k is a path from i to j alternating over the coloured edges c_k and $c_{k+1 \pmod p}$. Then, one can re-colour the edges by interchanging the alternating colours on each path. If the paths are disjoint, then the obtained edge-coloured graph is again a valid encoding for a schedule.

The concept of this neighbourhood for single-round robin tournaments can be generalised to double-round robin tournaments by replacing every edge in the graph by two anti-parallel arcs that can be coloured independently. That gives an arc-coloured complete directed graph, where the vertices correspond to the teams and the arc colours correspond to the rounds. The set of disjoint paths in the single-round robin case now translates to a set of arc-disjoint paths where it is permitted to traverse arcs in either direction, see Figure 1 for an example.

In the local search part of our procedure, we iterate over the possible disjoint paths, starting by checking small neighbourhoods: the sets of colours (rounds) of size 2. Once all possibilities are checked for all pairs of teams, this number of colours is increased. In each iteration we check the change of the objective, and based on this change we decide to accept the solution and restart, or to reject the solution and continue to the next iteration. We do this in a simulated annealing fashion, i.e. we also allow to continue with worse solutions with a certain probability. To ensure feasibility in Phase 3, solutions that violate hard constraints are never accepted. Finally, we terminate the procedure after a certain time limit and continue with the MIP approach.

3.2.2 MIP Approach

To improve on the schedule found by local search, we use a MIP model. Its main variables are $x_{i,j,r} \in \{0, 1\}$, where $i, j \in \mathcal{T}$ $(i \neq j)$ and $r \in \mathcal{S}$. This variable indicates that match (i, j) takes place in round r. The variables $hb_{i,r}, ab_{i,r} \in \{0, 1\}$ represent the home-break or away-break status of team i on slot $r \in \mathcal{S}$,



(a) Initial graph. (b) After re-colouring.

Fig. 1: An example of two neighbouring solutions using the colour-cycle (purple, blue, black) between team 3 and 6. The alternating paths are: purple–blue: $(3,5),(6,5)$; blue–black: $(2,3),(6,2)$; and black–purple: $(4,3),(4,6)$.

respectively. These are used to model the constraints related to breaks. For soft constraint class $c \in \mathcal{C}$, we denote by $d_{c,i}$ the penality of violating the i-th constraint in class c. The constraint penalty is denoted by p_c, and the model objective is to minimise the total penalty given by $\sum_{c\in\mathcal{C}} \sum_{i\in c} p_c d_{c,i}$. Each constraint is modelled by a set of linear constraints in a straightforward fashion, such that the feasible region matches the description of the constraint classes in the competition.

We use a fix-and-optimise matheuristic where we fix a (uniformly) random subset of rounds to the current schedule, and optimise for the remaining rounds. Given an initial feasible schedule, a random subset $\hat{\mathcal{S}} \subseteq \mathcal{S}$ is selected. Next, the variables x_{ijr} for $r \in \mathcal{S} \setminus \hat{\mathcal{S}}$ are fixed to the value corresponding to the initial solution, and the model is optimised. We have two parameters that change depending on the behaviour of the model: A time limit and a sample size. The same time limit parameter is used in the local search approach to ensure that the time spent in both approaches is about equal. If the time limit is reached while no improved solution is found, we increase the time limit and possibly reduce the sample size. If an optimal solution is found within the time limit, the random sample size is increased and the time limit is slightly decreased. Moreover, a solution limit is imposed. If multiple improved solutions are found within the time limit, we stop the optimisation and continue with another sample of rounds. In general, if after solving the MIP an improvement is found, we restart the MIP-approach with a different sample. Otherwise we go back to local search using the best schedule found so far as initialisation.

4 Computational Experience

For the implementation we have used Python 3.8. The MIPs are solved using Gurobi 9.1.0 with the `gurobipy` package. Table 1 compares the result of our solution approach (LS+MIP) with the approach where the compact MIP of Section 3.2.2 (MIP) is solved. The test set comprises the early test instances provided by [4]. Each instance has a time limit of 24 hours. A dash denotes that no feasible solution is found within the time limit. Observe that in most cases the solution process with local search finds a better primal solution than the compact MIP formulation. These results are found by using 4 cores on machines with an Intel Xeon Platinum 8260 with 42.7GB of RAM per instance.

Table 1: Computational results for the early instances comparing our method (LS+MIP) to solving the compact mixed-integer programming formulation (MIP) with a time limit of 24 hours per instance. Shown are the best primal solutions, dual bounds, the time per phase of LS+MIP and the number of iterations in LS and MIP.

	LS+MIP							MIP	
	Phase 2			Phase 3					
		Iterations			Iterations				
Instance	Time(s)	LS	MIP	Time(s)	LS	MIP	Primal	Primal	Dual
Early 1	822	6	15	85578	72	75	619	1588	1.00
Early 2	453	4	12	85947	57	94	369	606	0.00
Early 3	490	4	15	85910	50	69	1212	1701	55.19
Early 4	86400	60	98	–	–	–	–	–	0.00
Early 5	86400	65	101	–	–	–	–	–	258.88
Early 6	46127	60	82	40273	35	60	4682	–	633.64
Early 7	2049	12	25	84351	51	89	7306	10153	1271.64
Early 8	10	1	0	86390	60	91	1588	1615	210.40
Early 9	54	1	0	86346	82	121	443	206	0.00
Early 10	86400	50	79	–	–	–	–	–	319.77
Early 11	1647	17	26	84753	61	94	6772	10841	329.43
Early 12	61723	36	77	24677	31	41	1130	1850	0.00
Early 13	525	4	15	85875	53	81	372	498	2.00
Early 14	76	1	5	86324	54	114	24	42	1.00
Early 15	577	5	13	85823	59	78	4779	6021	514.48

References

1. Anderson, I.: Combinatorial designs and tournaments. 6. Oxford University Press (1997)
2. Briskorn, D., Drexl, A.: IP models for round robin tournaments. Computers & Operations Research **36**(3), 837–852 (2009)
3. Briskorn, D., Drexl, A., Spieksma, F.C.R.: Round robin tournaments and three index assignments. 4OR **8**(4), 365–374 (2010)
4. van Bulck, D., Goossens, D., Beliën, J., Davari, M.: The fifth international timetabling competition (ITC 2021): Sports timetabling. In: Proceedings of MathSport International 2021 Conference, MathSport, pp. 117–122 (2021)

5. Easton, K.K.: Using integer programming and constraint programming to solve sports scheduling problems. Ph.D. thesis, Georgia Institute of Technology (2003)
6. Goossens, D.R., Spieksma, F.C.R.: Soccer schedules in europe: an overview. Journal of scheduling **15**(5), 641–651 (2012)
7. Januario, T., Urrutia, S., Ribeiro, C.C., De Werra, D.: Edge coloring: A natural model for sports scheduling. European Journal of Operational Research **254**(1), 1–8 (2016)
8. Kendall, G., Knust, S., Ribeiro, C.C., Urrutia, S.: Scheduling in sports: An annotated bibliography. Computers & Operations Research **37**(1), 1–19 (2010)
9. Knust, S.: Classification of literature on sports scheduling. http://www.inf.uos.de/knust/sportssched/sportlit_class/. Accessed: June 2021
10. Nemhauser, G.L., Trick, M.A.: Scheduling a major college basketball conference. Operations research **46**(1), 1–8 (1998)
11. Rasmussen, R.V., Trick, M.A.: Round robin scheduling–a survey. European Journal of Operational Research **188**(3), 617–636 (2008)
12. Van Bulck, D., Goossens, D.: On the complexity of pattern feasibility problems in time-relaxed sports timetabling. Operations Research Letters **48**(4), 452–459 (2020)