First-break-heuristically-schedule: constructing highly-constrained sports timetables

David Van Bulck[a,b,*], Dries Goossens[a,b]

[a]*Ghent University, Faculty of Economics and Business Information*
[b]*FlandersMake@UGent – core lab CVAMO*

Abstract

Round-robin sports timetables are commonly generated in two phases: determining when teams play home and away, and when each team meets every other team. Depending on the phase solved first, the decomposition is known as first-break-then-schedule or first-schedule-then-break. This articles shows how Benders' decomposition regulating the home-away status of games in combination with variable neighbourhood search regulating the order of opponents explores the world between first-break-then-schedule and first-schedule-then-break. We use this framework to generate several new best solutions for ITC2021.

Keywords: Sports timetabling, International Timetabling Competition 2021 (ITC2021), Benders' decomposition, First-break-then-schedule, First-schedule-then-break, Matheuristics

1. Introduction

Sports timetables define who plays whom, when, and where. A special type of sports timetables are so-called k round-robin (kRR) timetables in which every team meets every other team exactly k times. In this paper, we focus on the popular double round-robin (2RR) format, where each team faces every other team twice: once as the home team, and once as the away team. More specifically, we are given a set of n teams T and set of time slots S. The 2RR timetable to be generated must be compact and the number of teams is assumed to be even, which implies that each team plays precisely one game per time slot (i.e. $|S| = 2(n-1)$). Apart from the basic requirement that each team plays at most one game at a time and that all games are scheduled, we are given a constraint set C, which is partitioned into hard (C_{hard}) and soft (C_{soft}) constraints. Hard constraints represent fundamental properties of the timetable that can never be violated, whereas soft constraints represent preferences that should be satisfied whenever possible. Real-life competitions usually feature hundreds of such constraints. The objective is to minimize the overall (weighted) sum of deviations from violated soft constraints while respecting all hard constraints.

Most contributions in the literature cope with these constraints by decomposing the timetabling problem into the following two subproblems, solved in either order, where the solution to one subproblem is used to constrain the other such that both solutions are compatible with each other and that they can merge into a timetable.

- Determine the opponent of each team in every time slot.
- Determine for each team in every time slot whether it plays home or away.

The solution to the former subproblem is referred to as the opponent schedule, whereas the solution to the latter subproblem is referred to as the home-away pattern (HAP) set.

When starting with the opponent schedule, the decomposition is known as 'first-schedule-then-break' (FSTB, see [1]). FSTB has been successfully used to schedule competitions where the objectives and constraints are mainly determined by the order of opponents (see e.g., [2, 3, 4]). On the other hand, FSTB does not seem very suitable when the objective or the constraints are mainly determined by the home-away status of games. Consider for instance the example of breaks; a team has a break whenever it plays consecutively at home or consecutively away. For the case of a 1RR with n teams, n even, Woeginger and co-authors [5, 6] show that, although a compatible HAP set always exists, fixing the opponent schedule may result in $O(n)$ times more breaks than strictly needed. Moreover, also from an algorithmic perspective there is little to gain as minimizing breaks for a given opponent schedule is $\mathcal{NP}$-hard. This is in sharp contrast to the celebrated result by de Werra [7], who shows how to solve the problem without fixed opponents in polynomial time.

When starting with the home-away patterns, the decomposition is known as 'first-break-then-schedule' (FBTS, see [8]). Although FBTS deals much better with breaks, the quality of the opponent schedule could be poor and it may even happen that no compatible assignment of opponents exists (see e.g., [9, 10]). Whenever a HAP set is infeasible, we thus need to backtrack. A first contribution in this regard is by Rasmussen and Trick [11], who propose to use logic-based Benders' cuts for backtracking. Although their approach offers a clear advantage over complete enumeration (which is surprisingly popular in practice), it assumes that the objective and almost all constraints are fully determined by the HAP set. A more versatile backtracking scheme based on traditional Benders' decomposition, was recently proposed by Van Bulck and Goossens

[*]Corresponding author
Email address: `david.vanbulck@ugent.be` (David Van Bulck)

This paper has been accepted for publication in OR Letters and is currently in press.

[12]. Nevertheless, their approach also has difficulty coping with large numbers of constraints and/or a complicated objective related to the order of opponents as it relies on integer programming (IP) to construct the opponent schedule.

This paper shows that a two-phase decomposition strategy is still appropriate when the timetable requirements can be more or less evenly attributed to both phases, a situation where neither FSTB nor FBTS would particularly perform well. For this, we first include a parameter in the Benders' decomposition algorithm by Van Bulck and Goossens [12] such that it allows to generate HAP sets that trade-off between satisfying wishes on the level of the home-away status of games versus satisfying wishes on the order of opponents. Subsequently, we show how to use the generated HAP sets as the input for a variable neighbourhood search algorithm that generates a compatible opponent schedule. We coin the resulting framework first-break-heuristically-schedule (FBHS), and show the effectiveness of FBHS by using it to generate several new best solutions for the benchmark instances used by the International Timetabling Competition 2021 on sports timetabling (ITC2021). This benchmark set is diverse, challenging, and realistic (see [13]), and at the same time well studied by the scientific community, and therefore highly suited to evaluate the performance of our proposed method.

The remainder of this paper is as follows. Section 2 outlines the proposed FBHS algorithm. Section 3 provides a description of problem instances used for the ITC2021 competition, and Section 4 provides computational results.

2. First-break-heuristically-schedule

The traditional first-break-then-schedule approach generates a sports timetable in two steps: first use IP to generate a HAP set (see Section 2.1), then use IP to generate a compatible opponent schedule or to prove that none exists (see Section 2.2). There are two issues with this approach, especially when there are several constraints on the level of the opponent schedule. First of all, it is very likely that the generated HAP set is infeasible or that the compatible opponent schedules are of poor quality. Section 2.3 explains how to solve this issue by separating traditional Benders' cuts. A second issue is related to the fact that finding a compatible opponent schedule could be too challenging for state-of-the-art IP solvers. Section 2.4 therefore introduces an IP-based variable neighbourhood search heuristic that is able to generate high-quality compatible opponent schedules within a reasonable amount of time.

2.1. Generate a HAP set

To generate a HAP set, we use the following IP formulation, where variables $h_{i,s}$ are 1 if team $i \in T$ plays home in time slot $s \in S$, and 0 if i plays away on s. The auxiliary variable $b^H_{i,s}$ ($b^A_{i,s}$) is 1 if team $i \in T$ has a home break (away break) in time slot $s \in S$, and 0 otherwise. Furthermore, denote with variable d_c the deviation of constraint $c \in C$, and with parameter p_c the weight of constraint $c \in C_{\text{soft}}$. Finally, we denote with $C^{\text{HAP}}_{\text{hard}} \subseteq C_{\text{hard}}$ and $C^{\text{HAP}}_{\text{soft}} \subseteq C_{\text{soft}}$ the hard and soft constraints that are fully determined by the HAP set, respectively.

Hap set generation model

$$\mathbf{min} \quad z_1 = \sum_{c \in C^{\text{HAP}}_{\text{soft}}} p_c d_c \tag{1}$$

$$\sum_{s \in S} h_{i,s} = (n-1) \qquad \forall i \in T \tag{2}$$

$$\sum_{i \in T} h_{i,s} = n/2 \qquad \forall s \in S \tag{3}$$

$$h_{i,s-1} + h_{i,s} - b^H_{i,s} + b^A_{i,s} = 1 \qquad \forall i \in T, s \in S \setminus \{1\} \tag{4}$$

$$b^H_{i,s} \leqslant h_{i,s},\; b^H_{i,s} \leqslant h_{i,s-1} \qquad \forall i \in T, \forall s \in S \setminus \{1\} \tag{5}$$

$$b^A_{i,s} + h_{i,s} \leqslant 1,\; b^A_{i,s} + h_{i,s-1} \leqslant 1 \qquad \forall i \in T, \forall s \in S \setminus \{1\} \tag{6}$$

$$f_c(h,b) - d_c \leqslant u_c \qquad \forall c \in C \tag{7}$$

$$d_c = 0 \qquad \forall c \in C^{\text{HAP}}_{\text{hard}} \tag{8}$$

$$d_c \geq 0 \qquad \forall c \in C^{\text{HAP}}_{\text{soft}} \tag{9}$$

$$h_{i,s}, b^H_{i,s}, b^A_{i,s} \in \{0,1\} \qquad \forall i \in T, \forall s \in S \tag{10}$$

The objective function minimizes the weighted deviation from constraints in $C^{\text{HAP}}_{\text{soft}}$. Constraints (2) state that each team plays half of its games at home, and Constraints (3) that exactly half of the teams in each time slot play home. Next, Constraints (4) to (6) regulate the value of the break variables. Constraints (7) model a set of application-specific constraints that involve the $h_{i,s}$ and auxiliary break variables (see Section 3). Constraints (8) and (9) state that hard constraints are strictly satisfied and that deviation from soft constraints are non-negative. Finally, Constraints (10) are the binary constraints.

2.2. Generate a compatible opponent schedule

Given a HAP set denoted by parameters $h'_{i,s}$, we now give another IP formulation which generates a compatible opponent schedule or proves that none exists. In this formulation, variables $x_{i,j,s}$ are 1 if team $i \in T$ plays at home against team $j \in T$ in time slot $s \in S$, and 0 otherwise.

Compatible opponent model

$$\mathbf{min} \quad \sum_{c \in C_{\text{soft}} \setminus C^{\text{HAP}}_{\text{soft}}} p_c d_c \tag{11}$$

$$\sum_{j \in T \setminus \{i\}} x_{i,j,s} = h'_{i,s} \qquad \forall\, i \in T, \forall\, s \in S \tag{12}$$

$$\sum_{j \in T \setminus \{i\}} x_{j,i,s} = 1 - h'_{i,s} \qquad \forall\, i \in T, \forall\, s \in S \tag{13}$$

$$\sum_{s \in S} x_{i,j,s} = 1 \qquad \forall i, j \in T : i \neq j \tag{14}$$

$$\sum_{s=1}^{n-1} (x_{i,j,s} + x_{j,i,s}) = 1 \qquad \forall i, j \in T : i < j \tag{15}$$

$$f_c(x) - d_c \leqslant u_c \qquad \forall c \in C \tag{16}$$

$$d_c = 0 \qquad \forall c \in C_{\text{hard}} \setminus C^{\text{HAP}}_{\text{hard}} \tag{17}$$

$$d_c \geq 0 \qquad \forall c \in C_{\text{soft}} \setminus C^{\text{HAP}}_{\text{soft}} \tag{18}$$

$$x_{i,j,s} \in \{0,1\} \qquad \forall i, j \in T : i \neq j, \forall\, s \in S \tag{19}$$

The objective function minimizes the weighted deviation from all soft constraints not yet considered in the HAP set generation model. Constraints (12) and (13) are the linking constraints, and require that all teams play according to their HAP. Constraints (14) ensure that all games are scheduled. If the timetable needs to be phased, we add Constraints (15). Constraints (16) represent a set of application-specific constraints that involve the $x_{i,j,s}$ variables (see Section 3). Constraints (17)

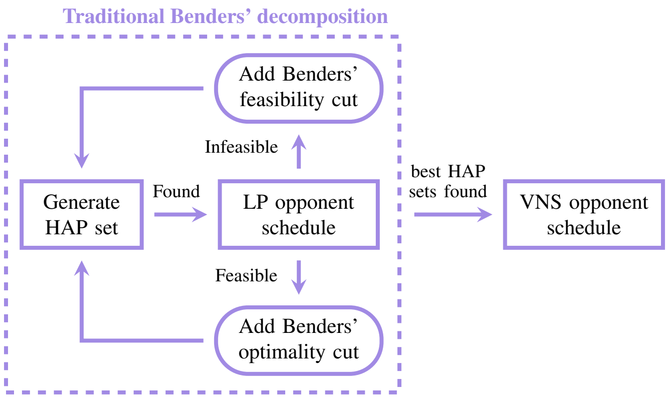

Figure 1: General structure of the proposed FBHS algoritm.

and (18) are similar to (8) and (9). Finally, Constraints (19) are the binary constraints on the $x_{i,j,s}$ variables.

2.3. Benders' decomposition

The traditional FBTS approach first solves the HAP set generation model and then the compatible opponent model, blindly repeating this sequence for the next best HAP set until a certain time or iteration limit is reached. As this method generates HAP sets without or with very little knowledge about the opponent schedule, the generated timetables – if any – are expected to be of poor quality.

Instead, this paper overcomes the aforementioned issue by strengthening the HAP set generation model with traditional Benders' cuts based on the linear programming (LP) relaxation of the compatible opponent model. The main idea of Benders' decomposition is that an IP model may become considerably easier once some complicating variables are fixed. Hence, Benders' decomposition first attempts to solve a so-called master problem that only considers the complicating variables after which it solves a subproblem that optimizes over all remaining variables. Provided that the subproblem is a linear program with fractional variables only, the contribution of Benders [14] is to show how linear duality theory can be used to iteratively add cuts to the master problem so as to ensure global feasibility and optimality.

In our approach, in line with Van Bulck and Goossens [12], we treat the variables in the HAP set generation model as the complicating variables and the variables in the LP-relaxation of the compatible opponent schedule as the continuous variables that we project out (see Figure 1). In other words, we first construct a HAP set by solving the HAP set generation model with traditional branch-and-bound techniques. For each integral HAP set encountered in the branch-and-bound tree, we then solve the LP relaxation of the subproblem. If the LP-relaxation turns out to be infeasible, we forbid the current and hopefully many other infeasible HAP sets by adding a Benders' infeasibility cut. On the other hand, if the LP-relaxation turns out to be feasible, we provide feedback about the lower bound on the quality of the opponent schedule by strengthening the HAP set generation model with a Benders' optimality cut.

As the Benders' decomposition model to generate HAP sets only considers the LP-relaxation of the opponent schedule, one may wonder how promising the generated HAP sets really are. For the case of a 1RR without any additional constraints on the order of opponents, Briskorn conjectures that feasibility of the LP-relaxation is in fact sufficient for the existence of a compatible opponent schedule. With regard to the quality of the opponent schedule, Van Bulck and Goossens [12] show that the generated lower bounds are as good as the Lagrangian relaxation relative to (12)-(13) or relative to (14). Interestingly, once the HAP set is fixed, the LP-relaxation based on the $x_{i,j,s}$ variables is also relaxation-equivalent to an exponentially-sized model where there is one variable for every possible perfect matching in the complete graph K_n (see van Doornmalen et al. [15]). The proof for this result is trivial, and follows from the fact that the well-known odd-set inequalities are redundant in a complete bipartite graph.

Nevertheless, even though the quality of the LP-relaxation looks promising, it remains a lower bound and hence the HAP set generation model is likely to be too optimistic about the quality of the opponent schedules that its resulting HAP set can yield. In order to provide a better trade-off between violated soft constraints in either of the two phases, we propose to multiply the objective function of the opponent model with parameter α. When $\alpha = 0$, the quality of the compatible opponent schedule is completely ignored, and hence the generated HAP sets will be similar to those generated with traditional FBTS. On the other hand, when α is very large, the generated HAP sets rather resemble those that would be found in the second phase of the FSTB approach. The desired value is somewhere in between, where the estimate corresponds to the quality of the best possible opponent schedule.

2.4. Variable neighbourhood search

Given the generated HAP sets, we now explain how to find compatible opponent schedules. Observing that half of the $x_{i,j,s}$ variables can be eliminated once the HAP set is fixed, Van Bulck and Goossens [12] propose the use of IP. Nevertheless, realistic problem instances typically feature a large number of hard and soft constraints on the order of opponents, making it intractable for an IP model to find (good) solutions in a reasonable time. Inspired by the winners of the ITC2021 competition (see [16]), we instead propose to create opponent schedules via a Variable Neighbourhood Search (VNS) matheuristic. One of the main advantages of matheuristics is that they directly operate on IP models, making it rather straightforward to include a wide variety of constraints.

Algorithm 1 outlines our proposed heuristic. In order to generate an initial feasible solution, we first ignore all hard and soft constraints and construct a compatible opponent schedule using the IP model from Section 2.2. Note that such opponent schedule always exists, unless Briskorn's conjecture is not valid for 2RR tournaments. Subsequently, we transform the hard constraints into soft constraints and we solve the resulting instance with the VNS approach. Once we obtain a zero-cost solution to the modified problem instance, we return to the original instance with all hard and soft constraints, and run the VNS again.

Algorithm 1 VNS – Compatible opponent schedule

```
$s_0 \leftarrow$ Solve model (12)-(15), (19)                 ▷ Ignore all hard and soft constraints
$s_1 \leftarrow$ VNS($s_0, \emptyset, C_{\text{hard}}$)      ▷ Add hard constraints as soft constraints
if GetObjective($s_1$) == 0 then
    $s_2 \leftarrow$ VNS($s_1, C_{\text{hard}}, C_{\text{soft}}$)   ▷ Consider all hard and soft constraints
end if

function VNS(startSol, hardCons, softCons)
    for $o$ in {Column, Row, Block} do
        $d = 4$
        do
            if ($o$ == Column) Select $d$ slots $S'$ and solve Column($S'$)
            if ($o$ == Row) Select $d$ teams $T'$ and solve Row($T'$)
            if ($o$ == Block) Select $d$ teams $T'$ and $d$ slots $S'$ and solve Block($T', S'$)
            if (500 iterations of while loop without improvement) $d = d + 1$
        while all subproblems solved to optimality
    end for

    $d = 4$
    do
        $e = 1$
        do
            Select $d$ teams $T'$ and $e$ slots $S'$ and solve RowsColumns($T', S'$)
            if (500 iterations of while loop without improvement) $e = e + 1$
        while all subproblems solved to optimality
        $d = d + 1$
    while $e > 1$        ▷ RowsColumns with $d - 1$ teams and 1 slot solved to optimality
end function
```

Team	s_1	s_2	s_3	s_4	s_5	s_6
1	2	4	3	4	2	3
2	1	3	4	3	1	4
3	4	2	1	2	4	1
4	3	1	2	1	3	2

Table 1: Overview of the different neighbourhoods in the VNS heuristic. The box in solid purple, dotted red, and dashed green represent Row({1, 2}), Column({2, 3}), and Block({3, 4}, {4, 5}), respectively. Neighbourhood RowColumn({1, 2}, {2, 3},) corresponds to the union of the solid purple and dotted red boxes.

The VNS heuristic repeatedly improves upon an incumbent solution by solving a series of smaller IP formulations that result from fixing some of the variables in the compatible opponent model. Let us denote with parameters $x'_{i,j,s}$ the incumbent solution from the previous iteration or the initial solution in case of the first iteration. We consider the following neighbourhoods (see also Table 1).

Column(S') The opponents related to time slots in $S' \subseteq S$ are free, all others are fixed. In other words, we set $x_{i,j,s} = x'_{j,i,s}$ for all $i, j \in T, i \neq j$, and $s \in S \setminus S'$.

Row(T') The opponents related to teams in $T' \subseteq T$ are free, all others are fixed. In other words, we set $x_{i,j,s} = x'_{i,j,s}$ for all $i, j \in T, i \neq j$ and $i \notin T'$ or $j \notin T'$, and $s \in S$.

Block(T', S') The opponents related to teams in $T' \subseteq T$ and time slots in $S' \subseteq S$ are free, all others are fixed. In other words, we set $x_{i,j,s} = x'_{i,j,s}$ for all $i, j \in T, i \neq j, s \in S \setminus S'$ and for all $i, j \in T \setminus T', s \in S'$.

RowColumn(T', S') The opponents either related to teams in $T' \subseteq T$ or time slots in $S' \subseteq S$ are free, all others are fixed. In other words, we set $x_{i,j,s} = x'_{i,j,s}$ for all $i, j \in T \setminus T', i \neq j$, $s \in S \setminus S'$.

While the first two operators are common, the third operator seems to be a natural extension. The fourth operator was introduced by Lamas-Fernandez et al. [16]. The neighbourhoods are sequentially searched in the above order, where we move on to the next neighbourhood as soon as one of the resulting subproblems could not be solved to optimality within a time limit of 120 seconds. Initially, the size of each neighbourhood is set to 4 teams and/or 4 time slot; the size of the neighbourhood is increased with 1 whenever no improvement was found in 500 iterations. Note that at each iteration, the VNS heuristic is warm-started with solution $x'_{i,j,s}$. The outcome of each iteration is thus never worse than the best solution found so far.

In order to select the specific teams and time slots used in each neighbourhood, we use duality theory. More in particular, whenever a new best solution $x'_{i,j,s}$ is found, we (temporally) add constraints $x_{i,j,s} \geq x'_{i,j,s}$ and compute the related dual prices $\delta_{i,j,s}$. If $\delta_{i,j,s}$ is negative, a solution without variable $x_{i,j,s}$, ceteris paribus, will result in an improvement of the objective function. Hence, to increase the likelihood that that costly $x_{i,j,s}$ variables are free to be optimized, the selection probability for team $i \in T$ (resp. slot $s \in S$) is set equal to $\sum_{j \in T \setminus \{i\}} \sum_{s \in S} (\delta_{i,j,s} + \delta_{j,i,s})$ (resp. $\sum_{i,j \in T: i \neq j} \delta_{i,j,s}$) divided by $\sum_{i,j \in T: i \neq j} \sum_{s \in S} \delta_{i,j,s}$.

3. The ITC2021 sports timetabling format

The ITC2021 competition, which ran from October 2020 to April 2021, offers a benchmark of 45 highly-constrained sports timetabling problem instances, which are divided into three classes depending on when the instances were released ('Early', 'Middle', and 'Late')[1]. The ITC2021 problem format includes the following constraints $c \in C$, which are grouped into four classes.

Capacity constraints The first set of constraints force a team to play home or away and regulate the total number of games played by a group of teams against a set of other teams during a period in time.

CA1 = CA1$^{\text{H}}$ ∪ CA1$^{\text{A}}$ limit the total number of home games (CA1$^{\text{H}}$) or away games (CA1$^{\text{A}}$) played by team i_c during time slots $S_c \subseteq S$ to at most u_c.

CA2 = CA2$^{\text{H}}$ ∪ CA2$^{\text{A}}$ limit the total number of home games (CA2$^{\text{H}}$) or away games (CA2$^{\text{A}}$) played by team i_c against teams in $T_c \subseteq T$ during time slots $S_c \subseteq S$ to at most u_c.

CA3 = CA3$^{\text{H}}$ ∪ CA3$^{\text{A}}$ limit the total number of home games (CA3$^{\text{H}}$) or away games (CA3$^{\text{A}}$) played by team i_c against teams in $T_c \subseteq T$ in each sequence of k_c time slots to at most u_c.

CA4 = CA4$^{\text{H}}$ ∪ CA4$^{\text{A}}$ ∪ CA4$^{\text{HA}}$ limit the overall number of home games (CA4$^{\text{H}}$) or away games (CA4$^{\text{A}}$) or home and away games (CA4$^{\text{HA}}$) played by a set of teams $T^1_c \subseteq T$ against a set of teams $T^2_c \subseteq T$ during time slots in $S_c \subseteq S$ to at most u_c.

[1]Problem instances can be downloaded at `www.itc2021.ugent.be`.

Break constraints Break constraints impose an upper limit on the total number of breaks for a team over a set of time slots, or impose a limit on the total number of breaks in the timetable. BR1 limit the total number of breaks of team i_c during time slots $S_c \subseteq S$ to at most u_c.

BR2 limit the total number of breaks over all teams and all time slots to at most u_c.

Game constraints The third class of constraints enforce or forbid specific assignments of games to time slots.

GA1 require that at least l_c and at most u_c games from $G_c \subseteq T \times T$ are scheduled in $S_c \subseteq S$.

Fairness and separation constraints The last set of constraints aims to increase the fairness and attractiveness of the tournament.

FA2 requires that the timetable is 2-ranking-balanced, meaning that the difference in played home games between any two teams at any point in time is at most 2.

SE1 requires that there are at least 10 time slots between any two games involving the same opponents.

Apart from these constraints, some problem instances also require that the timetable is 'phased', meaning that the matches in the first $n-1$ and the last $n-1$ time slots each define a 1RR.

Clearly, both phases in the decomposition relate to several constraints in the ITC2021 format. With respect to generating a HAP set, this means that Constraints (7) are specified as follows.

Hap set – ITC2021 constraints

$$\sum_{s \in S_c} h_{i_c,s} - d_c \leq u_c \qquad \forall c \in \mathrm{CA1^H} \quad (20)$$

$$\sum_{p=s}^{s+k_c-1} h_{i_c,p} - d_{c,s} \leq u_c \quad \forall c \in \mathrm{CA3^H} : T_c = T, \forall s \in S : s < |S| - k_c + 1 \quad (21)$$

$$\sum_{i \in T_c^1} \sum_{s \in S_c} h_{i,s} - d_c \leq u_c \qquad \forall c \in \mathrm{CA4^H} : T_c^2 = T \quad (22)$$

$$\sum_{s \in S_c} b^H_{i_c,s} + b^A_{i_c,s} - d_c \leq u_c \qquad \forall c \in \mathrm{BR1} \quad (23)$$

$$\sum_{i \in T} \sum_{s \in S} b^H_{i,s} + b^A_{i,s} - d_c \leqslant u_c \qquad \forall c \in \mathrm{BR2} \quad (24)$$

$$\sum_{p=1}^{s} (h_{i,p} - h_{j,p}) - d_{c,\{i,j\}} \leqslant 2 \qquad \forall i,j \in T : i \neq j, \forall s \in S, \forall c \in \mathrm{FA2} \quad (25)$$

Constraints (20) to (22) respectively model the $\mathrm{CA1^H}$ constraints, and the $\mathrm{CA3^H}$ and $\mathrm{CA4^H}$ constraints for which $T_c = T_c^2 = T$. The constraints to model the corresponding variants of the $\mathrm{CA1^A}$, $\mathrm{CA3^A}$ and $\mathrm{CA4^A}$ constraints are identical except that $h_{i_c,s}$ is replaced by $1 - h_{i_c,s}$. Constraints (23) and (24) regulate the BR1 and BR2 break constraints. Finally, Constraints (25) state that the timetable is two-ranking-balanced.

In order to model the constraints related to the order of opponents, Constraints (16) take the following form.

Opponent schedule – ITC2021 constraints

$$\sum_{j \in T_c} \sum_{s \in S_c} x_{i_c,j,s} - d_c \leq u_c \qquad \forall c \in \mathrm{CA2^H} \quad (26)$$

$$\sum_{j \in T_c} \sum_{p=s}^{s+k_c-1} x_{i,j,p} - d_{c,s} \leq u_c \qquad \forall c \in \mathrm{CA3^H} : T' \subset T, \forall s \in S : s < |S| - k_c + 1 \quad (27)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{s \in S_c} x_{i,j,s} - d_c \leq u_c \qquad \forall c \in \mathrm{CA4^H} : T_c^2 \subset T \quad (28)$$

$$l_c - d_c \leq \sum_{(i,j) \in G_c} \sum_{s \in S_c} x_{i,j,s} \leq u_c + d_c \qquad \forall c \in \mathrm{GA1} \quad (29)$$

$$\sum_{s=n}^{2(n-1)} s\,(x_{i,j,s} + x_{j,i,s}) - \sum_{s=1}^{n-1} s\,(x_{i,j,s} + x_{j,i,s}) + d_c >= 11 \qquad \forall c \in \mathrm{SE1}, \forall i,j \in T : i < j \quad (30)$$

Constraints (26) model the $\mathrm{CA2^H}$ constraints, while Constraints (27) and (28) model $\mathrm{CA3^H}$ and $\mathrm{CA4^H}$ for $T_c, T_c^2 \subset T$. The constraints for $\mathrm{CA2^A}$, $\mathrm{CA3^A}$, and $\mathrm{CA4^A}$ constraints are identical, except that $x_{i,j,s}$ is replaced by $x_{j,i,s}$. For the $\mathrm{CA4^{HA}}$ constraints, replace $x_{i,j,s}$ by $x_{i,j,s} + x_{j,i,s}$. Constraints (29) model the GA1 constraints. In case the timetable is phased, we use Constraints (30) to model SE1. In order to model the more general case, we make use of auxiliary variables $y_{i,j,s,t}$ that are one if and only if team $i \in T$ and team $j \in T$ play against each other in time slot $s \in S$ and $t \in S$ (see Van Bulck and Goossens [12] for a similar idea and more details).

4. Computational results

All formulations were solved by ILOG CPLEX 12.10, enabled with 10 cores for the HAP set generation and 1 core for the VNS heuristic. The HAP set generation model was solved a first time with $\alpha = 1$ and a time limit of 24 hours, and once more for $\alpha = 5$ and for $\alpha = 10$ with a time limit of 12 hours (using the solution of $\alpha = 1$ as warm start). For the best HAP set found with each of the three α levels, we subsequently ran the VNS heuristic with 50 different random seeds. A single run of the VNS heuristic required on average 1 hour and 45 minutes; as the VNS heuristic only requires one thread, runs were executed in parallel. Note that the ITC2021 competition did not impose any time limit, and consequently many participants granted their algorithms significant running times.

Table 2 summarizes our computational results. The first three columns provide the name of the problem instance, and the best known lower bound and solution value as retrieved from the competition website. Next, for the best solution found with each α, the table provides the quality related to the HAP set, the LP-relaxation of the compatible opponent model, and the overall quality. Note that this LP-relaxation multiplied with α corresponds to the quality of the opponent schedule as estimated by the Benders' optimality cuts in the HAP set generation model. The table shows that we can indeed trade off between the quality of the HAP set and the quality of the opponent schedule by changing α. For 34 out of 45 of the problem instances (about 80%), we found a feasible solution. Given the fact that finding finding feasible solutions is far from trivial (see Van Bulck and Goossens [13]) and that we only considered few HAP sets for which we solved the second phase, this is a remarkable result. We also find 10 new best solutions, even though over 15 internationally renowned research teams have already tackled these

	Best known		$\alpha = 1$			$\alpha = 5$			$\alpha = 10$		
Inst.	LB	Sol.	HAP	LP	Sol.	HAP	LP	Sol.	HAP	LP	Sol.
E1	1	362	0	120	701	0	120	701	0	105	716
E2	0	145	0	0	330	0	0	330	0	0	330
E3	49	992	340	113	1122	460	65	1212	620	66	1428
E4	0	507	0	130	✗	0	129	✗	0	127	✗
E5	270	3127	5	1224	✗	1	1165	✗	0	1224	✗
E6	607	3325	501	1524	✗	767	1349	✗	777	1285	✗
E7	1296	4763	0	2584	6070	0	2584	6144	0	2584	5801
E8	213	1051	20	730	1570	0	731	1595	55	722	1620
E9	0	108	40	0	**56**	40	0	**56**	40	0	**56**
E10	331	3400	0	1359	✗	0	1342	✗	49	1328	✗
E11	348	4426	1	1409	4577	1	1409	4577	1	1409	4577
E12	0	380	320	0	✗	300	0	✗	280	0	**315**
E13	2	121	0	2	227	1	1	213	1	1	212
E14	1	4	42	0	42	33	0	33	42	0	42
E15	485	3362	225	1528	**2955**	775	1322	3477	1115	1211	3807
M1	2955	5177	✗	✗	✗	✗	✗	✗	x	✗	✗
M2	2984	7381	10	4933	✗	9	4916	✗	9	4916	✗
M3	3378	9542	730	5168	✗	690	5064	✗	765	4964	✗
M4	7	7	0	7	64	0	7	42	5	6	58
M5	47	413	252	37	296	233	38	**279**	298	30	338
M6	24	1120	365	21	1825	405	0	1845	320	2	✗
M7	27	1783	437	830	3561	491	682	3208	437	830	3561
M8	2	129	0	2	258	0	2	279	0	2	289
M9	0	450	60	2	445	60	0	**415**	60	0	450
M10	4	1250	10	459	1777	1	437	1745	1	446	1842
M11	345	2446	0	1207	✗	0	1207	✗	0	1207	✗
M12	1	911	324	0	**674**	324	0	**674**	304	0	829
M13	0	252	0	0	1930	0	0	2020	0	0	1940
M14	0	1172	0	360	**1150**	0	360	1154	0	360	**1150**
M15	1	485	340	116	1224	340	116	1224	1105	12	1705
L1	1103	1922	58	1734	2206	58	1723	2265	58	1729	2265
L2	2818	5400	0	4098	5545	0	4066	5520	0	4041	5551
L3	416	2369	370	1044	2808	895	828	3129	995	830	3278
L4	0	0	0	0	350	0	0	321	0	0	321
L5	398	1923	0	1466	✗	9	1382	✗	0	1385	✗
L6	6	923	351	83	✗	409	52	✗	577	0	✗
L7	5	1558	4	454	1863	4	454	✗	4	454	1933
L8	78	934	326	24	1237	306	21	1202	382	13	1024
L9	3	563	80	45	513	100	11	**498**	140	12	589
L10	1	1945	0	525	✗	0	525	✗	0	525	✗
L11	0	202	0	0	446	0	0	446	0	0	446
L12	203	3428	9	1217	4059	9	1217	4059	9	1217	4059
L13	7	1820	334	246	2878	592	2	2816	544	4	2729
L14	7	1202	0	354	1242	0	315	**1174**	0	344	1243
L15	0	20	**0**	0	**0**	0	0	**0**	0	0	**0**

Table 2: Computational results for the ITC2021 problem instances. An x-mark means that no solution was found; new best found solutions are indicated in bold. During experimental testing, an even better solution with value of 599 was found for instance M12, and a solution of 1140 for M14.

benchmark instances. These results let us believe that decomposition schemes are still appropriate when the requirements of neither of the two phases dominate.

In memoriam: Prof. dr. Gerhard Woeginger

This article is dedicated to Prof. dr. Gerhard Woeginger, who passed away April 1, 2022. One of Gerhard's many research interests was sports: inspired by the composition of darts boards [17], the mathematics of playing golf [18], or the ranking of moody chess players [19], Gerhard always found a way to solve fundamental problems. In honour of Gerhard, the corresponding instance for 16 teams, showing how poorly FSTB can perform in terms of breaks, has been added to the RobinX repository (see `www.sportscheduling.ugent.be/RobinX/breakRepo.php`).

Acknowledgement

David Van Bulck is a postdoctoral research fellow funded by the Research Foundation – Flanders (FWO) [1258021N].

References

References

[1] M. A. Trick, A schedule-then-break approach to sports timetabling, in: E. Burke, W. Erben (Eds.), Pract. Theory Autom. Timetabling III, Springer, Berlin, Heidelberg, 2001, pp. 242–253.

[2] K. Cheung, Solving mirrored traveling tournament problem benchmark instances with eight teams, Discrete Optim. 5 (2008) 138–143.

[3] R. V. Rasmussen, M. A. Trick, The timetable constrained distance minimization problem, Ann. Oper. Res. 171 (2008) 45.

[4] D. Goossens, X. Yi, D. Van Bulck, Fairness trade-offs in sports timetabling, in: C. Ley, Y. Dominicy (Eds.), Science meets Sports: when statistics are more than numbers, Cambridge Publishing Scholars, 2020, pp. 213–244.

[5] G. Post, G. Woeginger, Sports tournaments, home-away assignments, and the break minimization problem, Discrete Optim. 3 (2006) 165 – 173.

[6] A. E. Brouwer, G. F. Post, G. J. Woeginger, Tight bounds for break minimization in tournament scheduling, J. Comb. Theory Ser. A 115 (2008) 1065–1068.

[7] D. de Werra, Scheduling in sports, in: P. Hansen (Ed.), Studies on graphs and discrete programming, North-Holland, Amsterdam, 1981, pp. 381–395.

[8] G. L. Nemhauser, M. A. Trick, Scheduling a major college basketball conference, Oper. Res. 46 (1998) 1–8.

[9] D. Briskorn, Feasibility of home-away-pattern sets for round robin tournaments, Oper. Res. Lett. 36 (2008) 283 – 284.

[10] D. Van Bulck, D. Goossens, On the complexity of pattern feasibility problems in time-relaxed sports timetabling, Oper. Res. Lett. 48 (2020) 452 – 459.

[11] R. V. Rasmussen, M. A. Trick, A Benders approach for the constrained minimum break problem, Eur. J. Oper. Res. 177 (2007) 198 – 213.

[12] D. Van Bulck, D. Goossens, A traditional benders' approach to sports timetabling, Eur. J. Oper. Res. 307 (2023) 813–826.

[13] D. Van Bulck, D. Goossens, The international timetabling competition on sports timetabling (ITC2021), Eur. J. Oper. Res. In press (2022) 1–18. URL: `doi.org/10.1016/j.ejor.2022.11.046`.

[14] J. Benders, Partitioning procedures for solving mixed-variables programming problems, Numer. Math. 4 (1962) 238–252.

[15] J. van Doornmalen, C. Hojny, R. Lambers, F. C. R. Spieksma, Integer programming models for round robin tournaments, 2022. URL: `doi.org/10.48550/arXiv.2210.08292`.

[16] C. Lamas-Fernandez, A. Martinez-Sykora, C. N. Potts, Scheduling double round-robin sports tournaments, in: P. De Causmaecker, E. Özcan, G. Vanden Berghe (Eds.), Proc. 13th Int. Conf. Pract. Theory Autom. Timetabling, volume 2, PATAT, 2021, pp. 435 – 448.

[17] V. G. Deineko, G. J. Woeginger, Some problems around travelling salesmen, dart boards, and euro-coins, Bull. Eur. Assoc. Theor. Comput. Sci. 90 (2006) 43–52.

[18] G. Rinaldi, U. Voigt, G. J. Woeginger, The mathematics of playing golf, or: a new class of difficult non-linear mixed integer programs, Math. Program. 93 (2002) 77–86.

[19] F. J. van de Bult, G. J. Woeginger, The problem of the moody chess players, Inf. Process. Lett. 108 (2008) 336–337.