On the complexity of pattern feasibility problems in time-relaxed sports timetabling

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

[a] *Ghent University, Faculty of Economics and Business Information*

Abstract

Time-relaxed sports timetables utilize more time slots than minimally needed to schedule all the games. We consider the decision problem whether a time-relaxed single round-robin tournament can be scheduled given a game-off-day pattern (GOP) or home-away pattern (HAP) set. Bao proposes necessary conditions for the HAP and GOP set feasibility and asks whether these conditions are sufficient. We answer this question negatively, analyze and propose necessary conditions, and show that the time-relaxed HAP and GOP set feasibility problems are $\mathcal{NP}$-complete.

Keywords: Time-relaxed sports scheduling, home-away pattern set feasibility, game-off-day pattern set feasibility, round-robin, bipartite

1. Introduction

This paper focuses on so-called time-relaxed single round-robin tournaments. In a single round-robin tournament (1RR), each team meets each other team exactly once. In contrast to time-constrained timetables, time-relaxed timetables utilize more time slots than minimally needed to schedule all the games. Despite the large amount of contributions for sports timetabling (see Van Bulck et al. [1] for an overview) only a small minority of these studies deal with time-relaxed timetabling (e.g., Schönberger et al. [2], Van Bulck et al. [3], Van Bulck and Goossens [4]). This is somewhat surprising since most non-professional competitions and several well known professional competitions (e.g., the National Basketball Association league, see Bao [5]) are in fact time-relaxed.

Formally, a 1RR consists of a set of teams T, $|T| = n$, and a set of time slots S. A 1RR is time constrained if $|S| = n - 1$ for n even or $|S| = n$ for n odd, otherwise it is time-relaxed. A feasible timetable assigns games (i, j), i.e. a home game of team i against team j, to time slots such that either game (i, j) or game (j, i) is planned, and each team plays at most one game per time slot.

A popular method to construct sports timetables is the *first-break-then-schedule* approach that breaks down the timetabling process into two sub-problems (see Nemhauser and Trick [6]). The first sub-problem is to determine for each team its *home-away pattern* (HAP). The HAP of team i is a function $h_i : S \to \{H, A, O\}$ such that $h_i(s) = H$ if i plays a home game, $h_i(s) = A$ if i plays an away game, and $h_i(s) = O$ if i has an off day (also called bye) on time slot s. Given a HAP for each team, the second sub-problem determines which opponent each team faces for each of the time slots. Clearly, the assignment of opponents must be compatible with the HAPs before they can merge into a timetable: for each pair of opponents, the corresponding home-away patterns need to give one team the home advantage, and designate an away game for the other team. Moreover, if a team does not play against any opponent, it must have an off day in its pattern.

A collection of n HAPs is known as a HAP set. The *HAP set feasibility problem* departs from a HAP set and asks whether there exists a compatible assignment of opponents. If the answer is 'yes', the pattern set is feasible. Note that with respect to feasibility, the order of the patterns or their assignment to teams does not matter and hence will be ignored in this paper.

Numerous researchers have stressed the importance of characterizing the feasibility of HAP sets. Nemhauser and Trick state that 'it would be interesting to have a characterization of feasible pattern sets or to show that no compact characterization exists by showing, for instance, that the question of determining if a set is feasible is $\mathcal{NP}$-complete [6, p. 8]'. Easton states that 'if there were a known set of necessary and sufficient conditions for feasible pattern sets, this would favorably affect the ratio of feasible to infeasible pattern sets, the number of pattern sets generated, and quite possibly the run time of the pattern set generation algorithm [7, p. 1143]'. More recently, the HAP set feasibility problem was presented in the 'open problem session' of MAPSP 2017 [8].

A team has a break when it plays two consecutive home games or two consecutive away games in a row. For time-constrained HAP sets having an even number of teams and the minimum number of breaks, Miyashiro et al. [9] conjecture that the HAP set feasibility problem can be solved in polynomial time. A stronger conjecture is made by Briskorn [10], who conjectures that the time-constrained HAP set feasibility problem can be solved in polynomial time when the number of teams is even. To date, these two conjectures are open. Goossens and Spieksma [11] generalize the concept of breaks by defining a break as two home games (or two away games) in a given pair of arbitrary time slots. They show that it is $\mathcal{NP}$-complete to

*Corresponding author

Email address: david.vanbulck@ugent.be (David Van Bulck)

This is the peer-reviewed author-version of https://doi.org/10.1016/j.orl.2020.05.005, *published in Operations Research Letters.*

Table 1: Complexity status of MinCost for different costs structures. New results are indicated in bold.

	Time-constrained	Time-relaxed
Binary costs	$\mathcal{NP}$-hard, $\notin \mathcal{APX}$ [7, 14]	**$\mathcal{NP}$-hard, $\notin \mathcal{APX}$ (trivial)**
Symmetric costs	$\mathcal{NP}$-hard [7]	**$\mathcal{NP}$-hard (trivial)**
GOP set feasibility	**$\mathcal{P}$ (trivial)**	**$\mathcal{NP}$-hard (Sec. 2)**
HAP set feasibility	n even: $\mathcal{P}$ Conjecture [10]	**$\mathcal{NP}$-hard (Sec. 3)**

decide whether a break-free HAP set exists.

The concept of HAPs is also used on the level of multiple leagues where interdependencies arise from teams in different leagues belonging to the same club. When each league contains the same even number of teams, Davari et al. [12] present a polynomial-time algorithm to assign each team to a HAP such that the total capacity violations over the clubs are minimized.

Similar to the first-break-then-schedule approach, Bao [5] proposes to break down the timetabling process into three sub-problems. The first sub-problem is to determine the *game-off-day patterns* (GOP). The GOP of team i is a function $g_i : S \to \{G, O\}$ such that $g_i(s) = G$ if i plays a game and $g_i(s) = O$ if i has an off day on time slot s. The second and third sub-problem are respectively to determine the HAP set and the opponents. Changing the order in which the different sub-problems are solved results in different decomposition schemes. However, note that the GOPs are fully defined once the HAP set or the assignment of opponents is known.

A GOP set consists of n not necessarily unique GOPs (i.e., the GOP set is a multiset). The *GOP set feasibility problem* departs from a GOP set and asks whether there exists a compatible assignment of opponents. Based on the necessary conditions for HAP set feasibility, Bao [5] provides three necessary conditions for GOP set feasibility and conjectures one of them to be sufficient (see Section 2.2).

The GOP set feasibility problem is related to the 'tournament scheduling problem with pre-fixed absences' (see Schauz [13]). In this problem, we are given a set of teams, a set of ordered time slots, and an equal number of pre-fixed absences per team (if a team is absent, it must have an off day). The problem is then to determine the minimal number of time slots such that the tournament can be scheduled.

Several sports timetabling problems define a profit or cost $c_{i,j,s}$ for scheduling game (i, j) on time slot s. As an example, $c_{i,j,s}$ can represent estimated ticket sales. In the minimum cost single round-robin tournament problem (MinCost), the problem is to construct a 1RR such that the sum of costs associated with the scheduled games is minimum.

Depending on the cost structure, we consider four special cases of MinCost (see Table 1). In the binary MinCost problem, the costs are restricted to $\{0, 1\}$. In the symmetric MinCost problem $c_{i,j,s} = c_{j,i,s}$ for all teams $i \in T, j \in T \setminus \{i\}$ and time slots $s \in S$. Easton [7] proves that it is $\mathcal{NP}$-complete to decide whether a time-constrained timetable exists when the opponents of some teams are fixed in given slots. This problem can be reduced to binary and symmetric MinCost by setting $c_{i,j,s} = c_{j,i,s} = 0$ and $c_{i,k,s} = c_{k,i,s} = c_{j,k,s} = c_{k,j,s} = 1$ for all $k \in T \setminus \{i, j\}$ whenever team i has to play team j, regardless of the home-away status, on time slot s. Hence, the binary and symmetric MinCost problems are both $\mathcal{NP}$-hard.

Briskorn et al. [14] give an alternative reduction from the planar three-index assignment problem (P3AP) to binary MinCost. Moreover, they show that no constant-factor approximation algorithm exists for binary MinCost, unless $\mathcal{P} = \mathcal{NP}$.

Since time-constrained timetabling is a special case of time-relaxed timetabling, the $\mathcal{NP}$-hardness results for the time-constrained case with binary and symmetric costs trivially hold for the time-relaxed case as well.

We can also model the GOP set feasibility problem using costs by setting $c_{i,j,s} = c_{j,i,s} = 0$ if team i and j have a G in their game-off-day pattern on time slot s, and to 1 otherwise. In this case, the costs are symmetric and $c_{i,k,s} = c_{k,i,s} = 1$ for all $k \in T$ if the GOP of i has an off day on time slot s. In time-constrained scheduling, the GOP set feasibility problem is trivially solvable. Indeed, if the number of teams is even, a GOP set is feasible if and only if no pattern contains an off day; if the number of teams is odd, a GOP set is feasible if and only if every pattern contains exactly one off day and no two patterns have their off day on the same time slot.

Finally, we can model the HAP set feasibility problem using costs by setting $c_{i,j,s} = 0$ if i's pattern contains an 'H' and j's pattern contains an 'A' on time slot s, and $c_{i,j,s} = 1$ otherwise. Alternatively, we can model the time-constrained HAP set feasibility problem with n even such that $c_{i,j,s} = c_{i,s}$ for all teams i and j and time slots s. Indeed, note that each team plays once in every time slot of a time-constrained 1RR with n even. Hence, teams play away if they do not play home. Therefore, we can ignore the 'A's in the pattern by setting $c_{i,j,s} = 0$ for all $j \in T$ if $i \in T$ has an 'H' on time slot $s \in S$, and $c_{i,j,s} = 1$ otherwise. The complexity status of this special case of MinCost is unknown as the reduction given in [14] uses a different cost structure. Moreover, it seems not straightforward to adapt the proof since any P3AP instance with $c_{i,j,s} = c_{i,s}$ has constant objective value $\sum_{i \in T} \sum_{s \in S} c_{i,s}$ and is therefore trivially solvable (see Burkard et al. [15, p. 314]).

The contributions and remainder of this paper are as follows. First, Section 2 shows that the time-relaxed GOP set feasibility problem is $\mathcal{NP}$-complete. Unless $\mathcal{P} = \mathcal{NP}$, we can therefore reject the conjecture by Bao [5]. Nevertheless, we show that the problem can be solved in polynomial time when each team has at most one off day, and we conjecture the problem to be polynomially solvable when the number of off days per team is either two or three. Section 3 proves that the time-relaxed HAP set feasibility problem is $\mathcal{NP}$-complete. Moreover, Section 3 proves the conjecture of Briskorn [10] for up to eight teams in a time-constrained setting, and provides an example showing that the conjecture does not hold for time-relaxed HAP sets. An overview of the complexity results is given in Table 1.

2. The game-off-day pattern set feasibility problem

In order to verify whether a GOP set is feasible, Bao [5] proposes Integer Programming (IP) formulation (1)-(4). In this formulation, $x_{i,j,s}$ is a binary variable that equals 1 if and only if

team i and j, $i < j$, compete on time slot s. The objective function maximizes the number of scheduled games. Constraints (1) restrict each pair of teams to meet at most once, whereas constraints (2) restrict each team to play at most one game per time slot. Finally, constraints (3) model the given GOP set, and constraints (4) are the binary constraints. A given GOP set is feasible if and only if $z_G = \binom{n}{2}$. We refer to formulation (1)-(4) as IP-GOP.

$$\max\ z_G = \sum_{i,j\in T:i<j} \sum_{s\in S} x_{i,j,s}$$

$$s.t. \sum_{s\in S} x_{i,j,s} \le 1 \qquad \forall i,j \in T : i < j \quad (1)$$

$$\sum_{j\in T:i<j} x_{i,j,s} + \sum_{j\in T:i>j} x_{j,i,s} \le 1 \qquad \forall i \in T, s \in S \quad (2)$$

$$x_{i,j,s} = 0 \qquad \forall i,j \in T : i < j, \forall s \in S : g_i(s) = O \vee g_j(s) = O \quad (3)$$

$$x_{i,j,s} \in \{0,1\} \qquad \forall i,j \in T : i < j, s \in S \quad (4)$$

As solving IP-GOP can be time-consuming, the question arises whether infeasible GOP sets can be detected more efficiently.

2.1. Complexity status of the GOP set feasibility problem

Theorem 1 shows that the time-relaxed GOP set feasibility problem is $\mathcal{NP}$-complete when each team has at least n off days.

Theorem 1. *The time-relaxed GOP set feasibility problem is $\mathcal{NP}$-complete, even if $|S| = 2n - 1$.*

Proof. We prove the theorem by presenting a reduction from constructing a time-constrained 1RR with fixed opponents (C-1RR).

C-1RR. *Instance*: Set U of teams with $|U|$ even, set P of time slots with $|P| = |U| - 1$, and a set A of fixed opponent assignments consisting of pairs $(\{u, v\}, p)$ meaning that team $u \in U$ has to play a game, regardless of the home-away status, against team $v \in U \setminus \{u\}$ on time slot $p \in P$.

Question: Is there a (time-constrained) timetable such that (u, v) or (v, u) is scheduled on time slot p whenever $(\{u, v\}, p) \in A$? Easton [7] proves that C-1RR is $\mathcal{NP}$-complete, even if A contains on average only two fixed opponents per team (i.e., $|A| = |U|$; recall that each entry in A requires two teams).

We now construct an instance of GOP from any instance of C-1RR. To begin, we set $T = U$ and add a set of time slots $S = S_1 \cup S_2$ with $S_1 = P$ and for each pair $(\{u, v\}, p) \in A$ a time slot $s_{\{u,v\}} \in S_2$. Finally, we construct the game-off-day pattern set as follows. For each time slot $s \in S_1$ we indicate that team $u \in U$ plays a game whenever it has no fixed opponent in s, and that u has an off day otherwise. In addition, for each time slot $s_{\{u,v\}} \in S_2$, we indicate that team u and v play a game, whereas all other teams have an off day. In summary, the corresponding instance of GOP is completely specified by:

$$T = U,$$
$$S = P \cup S_2 \Rightarrow |S| = |U| - 1 + |U| = 2n - 1,$$
$$g_i(s) = \begin{cases} \text{G} & \text{if } (\nexists j \in T \setminus \{i\} : (\{i,j\}, s) \in A)\ \forall i \in T, s \in S_1, \\ \text{O} & \text{if } (\exists j \in T \setminus \{i\} : (\{i,j\}, s) \in A)\ \forall i \in T, s \in S_1, \\ \text{G} & \text{if } (i = u \vee i = v)\ \forall i \in T, s_{\{u,v\}} \in S_2, \\ \text{O} & \text{if } (i \neq u \wedge i \neq v)\ \forall i \in T, s_{\{u,v\}} \in S_2. \end{cases}$$

We now show that the C-1RR instance admits a feasible solution if and only if the GOP instance admits a feasible solution. When given a feasible C-1RR solution, we construct a feasible timetable for the GOP instance as follows. We schedule game (i, j) on time slot s if and only if game (i, j) is scheduled on time slot s in the C-1RR solution and if the game between i and j is not fixed, i.e., $(\{i, j\}, s) \notin A\ \forall s \in P$. If the game between i and j is fixed, we schedule game (i, j) on time slot $s_{\{i,j\}}$ in the solution of the GOP instance. By construction $s_{\{i,j\}} \in S_2$ always exists, which makes that all games are scheduled. In a time-constrained timetable with an even number of teams, each team plays exactly one game per time slot. This makes that the patterns of teams in S_1 are respected since our construction method does not schedule any game for a team on time slot $s \in S_1$ if this team has a fixed opponent on s (i.e., the team must have an off day on s). Instead, we use the dedicated time slot in S_2 to schedule this game, which makes that the patterns in S_2 are also respected.

Conversely assume that we are given a feasible solution to the GOP instance. Then, we create a feasible solution for the C-1RR instance by first copying all game assignment in S_1. From the feasibility of the GOP solution it follows that this will result in at most one game per team. To deal with the fixed opponents, we note that time slot $s_{\{i,j\}} \in S_2$ must contain a game between team i and team j since these are the only teams that do not have off days in their pattern on $s_{\{i,j\}}$. Contrarily, the pattern of both teams contains an off day on the fixed time slot on which they have to meet each other. Without generating any conflict, we can therefore schedule either game (i, j) or (j, i) on time slot s whenever $(\{i, j\}, s) \in A$. □

2.2. Necessary conditions for GOP set feasibility

Unless $\mathcal{P} = \mathcal{NP}$ and provided that each team has at least n off days, Theorem 1 implies that no necessary and sufficient condition exists that can be checked in polynomial time. In this section, we explore whether efficient conditions exist when the number of off days per team is less than n.

Consider first the relation between a 1RR with one off day per team and a symmetric Latin square L, i.e. an $n \times n$ array filled with symbols $\{1, \ldots, n\}$ in such a way that each symbol occurs once in every row and column and such that $L_{i,j} = L_{j,i}$.

Lemma 1. *Every symmetric Latin square generates a 1RR with one off day per team which is unique up to the home advantage of the games, and every 1RR with one off day per team generates a unique symmetric Latin square.*

Proof. Given a symmetric Latin square L with symbol $L_{i,j}$ contained in the cell of the i-th row and j-th column, we construct a 1RR as follows. First, we assign an off day to team $i \in T$ on symbol $L_{i,i}$. For each $i, j \in \{1, \ldots, n\}$ with $i < j$, we then schedule either game (i, j) or (j, i) on symbol $L_{i,j} = L_{j,i}$. This

makes that all games are scheduled and that each team has exactly one off day. Since each symbol occurs at most once in every row and column, each team plays at most once per time slot. Moreover, it follows from the construction that the timetable is unique up to the home advantage of the games.

Given a 1RR with one off day per team, we generate a unique symmetric Latin square as follows. Let s_i be the unique time slot on which $i \in T$ has its off day. We then construct the principal diagonal by assigning s_i to $L_{i,i}$ for each $i \in T$. Furthermore, for each pair of teams $i, j \in T$ with $i < j$, we assign the time slot on which i plays against j to cell $L_{i,j}$ and $L_{j,i}$. This makes that the Latin square is symmetric and that all cells are filled. Moreover, since each team plays at most one game per time slot and has one off day, each element occurs exactly once per row and column. □

Condition 1 (Bao [5])**.** *The number of 'G's is even on each time slot. Equivalently, the number of 'O's is even (odd) whenever n is even (odd) on each time slot.*

Since each game involves two teams, Condition 1 is clearly necessary. Its simplicity notwithstanding, Condition 1 is also sufficient when each team has at most one off day.

Lemma 2. *When each team has one off day, the GOP set is feasible if and only if Condition 1 holds.*

Proof. It follows from Lemma 1 that a GOP with one off day per team is feasible if and only if we can construct a symmetric Latin square in which the elements on the diagonal correspond with the off days of the teams. Chang [16] proves that a symmetric $n \times n$ Latin square with a prescribed diagonal exists if and only if the diagonal contains each symbol exactly once for odd n, and an even number of times (possibly 0) for even n. □

In a *bipartite 1RR*, the set of teams T can be partitioned into two equally sized groups T_1 and T_2 such that teams never play against teams in their own group, and once against each team from the other group. A time-constrained bipartite 1RR is therefore equivalent to a Latin square L in which $L_{i,j}$ contains the time slot on which team $i \in T_1$ plays against $j \in T_2$. Interestingly, a 1RR between $n = 2m$ teams can be seen as a 1RR between the first m teams, a 1RR between the last m teams, and a bipartite 1RR between the first and the last m teams. Furthermore, we observe that permuting the columns (time slots) of a pattern set does not change the feasibility of a GOP set. We make use of these observations to prove the following theorem.

Theorem 2. *A timetable compatible with a GOP set in which each team has one off day can be constructed in $O(n^2)$ time if and only if the GOP set satisfies Condition 1.*

Proof. The construction is trivial when the number of teams is odd. In order to prove the theorem when the number of teams is even, we transform a proof by induction for the existence of a symmetric Latin square with a prescribed diagonal (see Chang [16]) into a recursive algorithm that is able to construct a timetable when given a set of $n = 2m$ teams, a set of n time slots, and a GOP set that satisfies Condition 1.

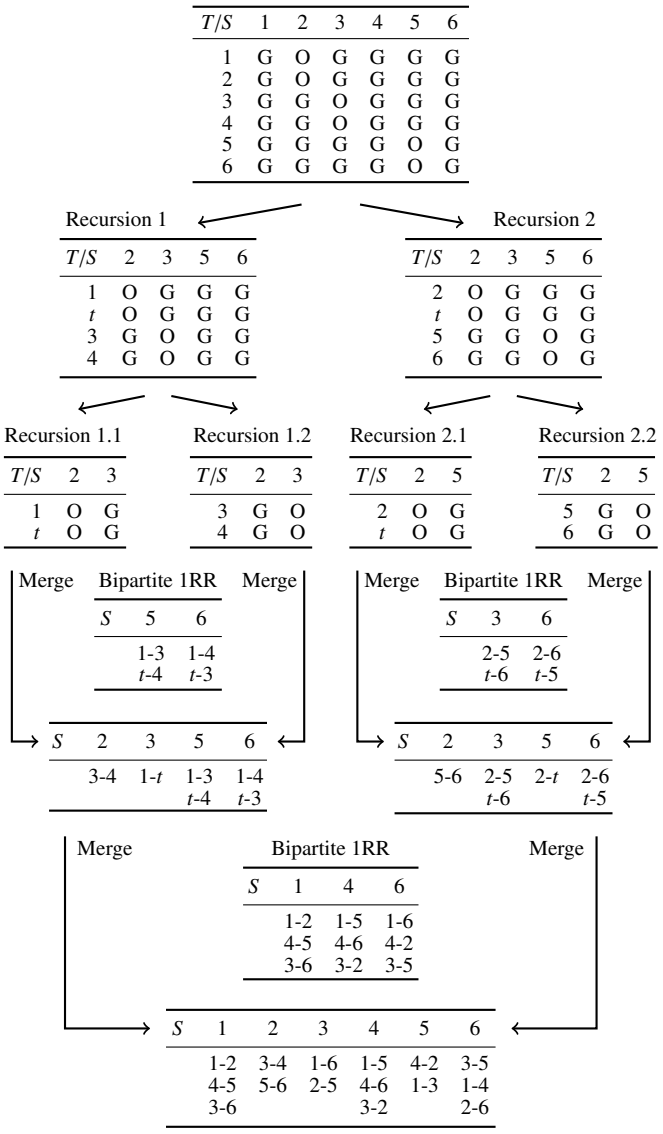

Figure 1: Illustration of the recursive algorithm for $n = 6$. *Recursion 1-2:* splitting into GOP sets with $n = 4$ using dummy team t, $s' = 2$, $s'' = 6$, $a_2 = a_3 = b_2 = b_5 = 2$, and $a_5 = b_3 = 0$. *Recursion 1.1-2.2:* a timetable for a GOP set with $n = 2$ is trivially constructed. *First level of merging:* a timetable with 4 teams and $\pi_1 = (1\ 4\ 3)$, $\pi_2 = (6\ 2\ 5)$. *Second level of merging:* rescheduling a bipartite 1RR in which the i-th team in π_1 plays against the i-th team in π_2 during time slot s'' results in a 1RR compatible with the initial GOP set.

Basis. If $n = 2$, we construct a timetable by scheduling the single game on the time slot without off days. If the given GOP set contains $2m$ teams with $m > 1$, we recursively construct the timetable as follows. First, let d_s represent the number of teams that have an off day on time slot s and denote with S^+ the subset of time slots for which $d_s > 0$. Depending on whether m is odd or even, we consider two different recursive calls.

Recursion m even. When m is even, we define even numbers a_s and b_s for each $s \in S^+$, such that $d_s = a_s + b_s$ and $m = \sum_{s \in S^+} a_s = \sum_{s \in S^+} b_s$. If d_s is a multiple of 4, we set $a_s = b_s = d_s/2$. Otherwise, we alternately set $a_s = d_s/2 - 1$ and $b_s = d_s/2 + 1$ or $a_s = d_s/2 + 1$ and $b_s = d_s/2 - 1$. We observe that it follows from the construction that $\sum_{s \in S^+} a_s = \sum_{s \in S^+} b_s$ since

there must be an even number of time slots for which d_s is not a multiple of 4. Indeed, the assumption that n and m are even implies that $m = \sum_{s \in S^+} a_s$ must be a multiple of 4. Next, we partition the time slots into two equally sized subsets S_1 and S_2 such that $S^+ \subseteq S_1$ and we set $a_s = b_s = 0$ for all $s \in S_1 \setminus S^+$ (Condition 1 guarantees that $|S^+| \leq m$).

We now partition T into two equally sized subsets T_1 and T_2 such that the number of teams in T_1 (resp. T_2) with an off day on time slot s is equal to a_s (resp. b_s), $\forall s \in S_1$. This is always possible since each team has exactly one off day.

In order to generate a (partial) timetable compatible with the associated GOP set for the teams in T_1 (resp. T_2) during the time slots in S_1 we make a recursive call. Finally, we construct in $O(m^2)$ time a Latin square representing a time-constrained bipartite 1RR between the teams in T_1 and the teams in T_2 during the time slots in S_2. This makes that all games of the 1RR are timetabled, that the original GOP set is respected, and that no team plays more than one game per time slot.

Recursion m odd. When m is odd, there must be at least one d_s, say $d_{s'}$, such that $d_{s'} = 4p - 2$. If we set $a_{s'} = b_{s'} = 2p$, we can use the approach described above to find even numbers a_s and b_s such that $d_s = a_s + b_s$ for each $s \in S^+ \setminus \{s'\}$, and such that $m + 1 = \sum_{s \in S^+} a_s = \sum_{s \in S^+} b_s$. Next, we partition the time slots into two equally sized subsets S_1 and S_2 such that $S^+ \subseteq S_1$ and we set $a_s = b_s = 0$ for all $s \in S_1 \setminus S^+$.

Denote with t an additional dummy team which has its off day on time slot s'. We then partition T into two equally sized subsets T_1 and T_2 such that the number of teams in $T_1 \cup \{t\}$ (resp. $T_2 \cup \{t\}$) with an off day on time slot $s \in S_1$ equals a_s (resp. b_s). The inclusion of t makes that $T_1 \cup \{t\}$ and $T_2 \cup \{t\}$ contain an even number of teams but requires that we include one more time slot: denote with s'' an arbitrary time slot in S_2.

In order to generate a (partial) timetable compatible with the associated GOP set for the teams in $T_1 \cup \{t\}$ (resp. $T_2 \cup \{t\}$) during the time slots in $S_1 \cup \{s''\}$ we can now make a recursive call. Denote with π_1 (resp. π_2) the sequence in which t plays against the teams in T_1 (resp. T_2). After removing all games involving t, we then get a timetable in which the i-th team in π_1 and π_2 does not play any game on the i-th time slot in $(S_1 \setminus \{s'\}) \cup \{s''\}$. We now generate a time-constrained bipartite 1RR between T_1 and T_2 during the time slots in S_2 such that the i-th team in π_1 plays against the i-th team in π_2 on time slot s''. After rescheduling the game between the i-th team in π_1 and the i-th team in π_2 on the i-th time slot in $S_1 \setminus \{s'\} \cup \{s''\}$, we get a timetable that respects the GOP set, in which all games are scheduled, and in which no team plays more than one game per time slot. An overview of the construction is given in Figure 1.

Running time. Let $T(n)$ denote the running time of the recursive algorithm on a problem of size n. The base case $T(2)$ is easy and requires constant time. The recursive case occurs when $n > 2$ and generates two sub-problems each of size $n/2$ and thus contributes by $2T(n/2)$. The construction of the bipartite tournament between the teams in T_1 and T_2 can be done by constructing a Latin square of order $n/2$ and requires $O\left((n/2)^2\right)$ time. Combining the running times, we get $T(n) = 2T(n/2) + O\left((n/2)^2\right)$. It follows from the master theorem for recurrence relations (see Cormen et al. [17, third case]) that the running time is dominated by the merging part: $T(n) = O\left((n/2)^2\right)$. □

Table 2: Example of an infeasible GOP set for $n = 4$ and two off days per team, satisfying Condition 1 but violating Condition 2.

T/S	1	2	3	4	5
1	G	G	G	O	O
2	G	G	G	O	O
3	G	O	O	G	G
4	G	O	O	G	G

Table 2 shows that Condition 1 is no longer sufficient when there are two off days per team: team 1 and team 2 are the only two teams that have a 'G' in their pattern on time slot 2 and 3. Hence, the GOP set is infeasible as team 1 and 2 can play at most one game against each other in a 1RR. We therefore propose Condition 2, which expresses for each pair of teams that there can be at most one time slot on which the two teams exclusively have a 'G' in their pattern.

Condition 2. *For each $i, j \in T : i \neq j$, there is at most one time slot $s \in S$ on which $g_i(s) = g_j(s) = G$ and $g_k(s) = O$ for $k \in T \setminus \{i, j\}$.*

Lemma 3. *The combination of Conditions 1 and 2 is necessary and sufficient when each team has two off days and there are exactly two off days on each but one time slot.*

Proof. The necessity of Conditions 1 and 2 was argued above. If each team has two off days, we observe that a 1RR corresponds to a symmetric quasi-Latin square (SQLS) in which the diagonal elements contain two symbols and all the off-diagonal elements contain one symbol. Under the conditions of the lemma, the GOP set feasibility problem is equivalent to determining whether an SQLS can be completed when all elements on the principal diagonal are filled such that each but one symbol appears twice on the diagonal. Bryant and Rodger [18, Lemma 2.8] prove that an SQLS of this form can always be completed, unless the number of rows is odd or the number of rows equals 4 and the diagonal contains two pairs with the same symbols. When the number of teams is odd, Condition 1 is violated as there are n time slots with two off days. When there are four teams and two teams are the only ones to have off days on two time slots, Condition 2 will be violated. For the setting described in this lemma, Conditions 1 and 2 are therefore necessary and sufficient. □

To stimulate further research, we formulate Conjecture 1.

Conjecture 1. *When each team has exactly two off days, the combination of Conditions 1 and 2 is a sufficient condition.*

In order to verify Conjecture 1 for given n, we use a constraint programming solver (CPLEX CP Optimizer) to enumerate all GOP sets satisfying Condition 1 and check whether either Condition 2 is violated or $z_G = \binom{n}{2}$ (IP-GOP solved with

Table 3: Example of an infeasible GOP set for $n = 6$ and three off days per team, satisfying Conditions 1 and 2 but violating Condition 3 ($\bar{z}_G = 13$).

T/S	1	2	3	4	5	6	7	8
1	O	O	G	G	G	G	O	G
2	O	O	G	G	G	O	G	G
3	G	G	G	G	G	O	O	O
4	G	G	O	O	G	G	G	O
5	G	G	O	G	O	G	G	O
6	G	G	G	O	O	G	G	O

Table 4: Example of an infeasible GOP set for $n = 4$ and one off day per team, violating Condition 1 but satisfying Condition 3 ($\bar{z}_G = 6$). In the LP-relaxation, we have $x_{1,3,1} = x_{2,3,1} = x_{1,4,1} = x_{2,4,1} = x_{1,2,2} = x_{1,3,2} = x_{2,3,2} = x_{1,2,3} = x_{1,4,3} = x_{2,4,3} = 0.5$ and $x_{3,4,4} = 1$.

T/S	1	2	3	4
1	G	G	G	O
2	G	G	G	O
3	G	G	O	G
4	G	O	G	G

CPLEX). Following this procedure, we are able to confirm validity of Conjecture 1 for $n \leq 10$. For $n > 10$, the number of GOP sets becomes too large to enumerate and check in a reasonable amount of time. Instead, we use the constraint programming solver in combination with a multistart search strategy to generate and check 10,000 GOP sets for each value $n \in \{11, 12, \ldots, 20\}$. For each of the 100,000 additionally considered GOP sets, Conjecture 1 could not be rejected.

When there are three or more off days per team, Table 3 shows that the combination of Conditions 1 and 2 is not sufficient.

Bao [5] derives Condition 3 based on the Linear Programming (LP) relaxation of IP-GOP.

Condition 3 (Bao [5])**.** *Denote with $\bar{z}_G$ the objective value of the LP-relaxation of IP-GOP, then $\bar{z}_G = \binom{n}{2}$.*

Since $z_G \leq \bar{z}_G$ and since a given GOP set is feasible if and only if $z_G = \binom{n}{2}$, Condition 3 is a necessary condition that can be computed in polynomial time. Bao [5] conjectures that Condition 3 is also sufficient.

Conjecture 2 (Bao [5, p.57])**.** *Condition 3 is sufficient for GOP set feasibility.*

It follows from Theorem 1 that Conjecture 2 cannot hold, unless $\mathcal{P} = \mathcal{NP}$. Table 4 confirms the former as Condition 3 does not detect a violation of Condition 1 which is a necessary condition.

Denote with $c_G(T', s)$ the number of 'G's in the GOPs of a subset of teams $T' \subseteq T$ on time slot $s \in S$. In order to guarantee that violations of Condition 1 are detected, we add constraints (5).

$$\sum_{i,j \in T: i<j} x_{i,j,s} \leq \frac{c_G(T,s) - 1}{2} \qquad \forall s \in S : c_G(T,s) \bmod 2 = 1 \quad (5)$$

We propose a weaker version of Conjecture 2.

Table 5: Example of an infeasible GOP set for $n = 6$ and four off days per team, satisfying Conditions 1 to 4 ($\bar{z}_G = 15$). In the LP-relaxation, we have $x_{1,2,1} = x_{1,6,1} = x_{2,5,1} = x_{5,6,1} = x_{3,4,2} = x_{3,6,2} = x_{4,5,2} = x_{5,6,2} = x_{1,2,3} = x_{1,3,3} = x_{2,4,3} = x_{3,4,3} = x_{2,3,4} = x_{2,4,4} = x_{3,6,4} = x_{4,6,4} = x_{1,5,5} = x_{1,6,5} = x_{4,5,5} = x_{4,6,5} = x_{1,3,6} = x_{1,5,6} = x_{2,3,6} = x_{2,5,6} = 0.5$ and $x_{1,4,7} = x_{2,6,8} = x_{3,5,9} = 1$.

T/S	1	2	3	4	5	6	7	8	9
1	G	O	G	O	G	G	G	O	O
2	G	O	G	G	O	G	O	G	O
3	O	G	G	G	O	G	O	O	G
4	O	G	G	G	G	O	G	O	O
5	G	G	O	O	G	G	O	O	G
6	G	G	O	G	G	O	O	G	O

Conjecture 3. *When each team has exactly three off days, the combination of Condition 3 strengthened with constraints* (5) *is a sufficient condition.*

By enumerating all possible GOP sets with constraint programming (CPLEX CP Optimizer), to permutations of the order of the time slots, and by checking whether Condition 3 strengthened with constraints (5) detects infeasibility, we answer Conjecture 3 affirmatively for $n \leq 8$. For $n > 8$, the number of GOP sets becomes too large to enumerate and check in a reasonable amount of time. Instead, we use the constraint programming solver in combination with a multistart search strategy to generate and check 10,000 GOP sets for each value $n \in \{9, 10, \ldots, 20\}$. For each of the 120,000 additionally considered GOP sets, Conjecture 3 could not be rejected.

Clearly, the number of games between teams in T' on s is at most $\lfloor \frac{c_G(T',s)}{2} \rfloor$. Hence, Condition 4 is a necessary condition that requires to check $O(2^n)$ number of constraints.

Condition 4 (Bao [5])**.** $\sum_{s \in S} \left\lfloor \frac{c_G(T',s)}{2} \right\rfloor \geq \binom{|T'|}{2}$ *for each* $T' \subseteq T$.

One can wonder whether the combination of Conditions 1 to 4 is sufficient when there are four off days per team. Table 5 shows this is not the case. Note that Table 5 also proves that Conjecture 2 does not hold for pattern sets that satisfy Condition 1.

3. The home-away pattern set feasibility problem

In order to verify whether a HAP set is feasible, Briskorn [10] proposes the following IP formulation, referred to as IP-HAP.

$$\max \; z_H = \sum_{i,j \in T: i<j} \sum_{s \in S} x_{i,j,s}$$

$$s.t. \; (1), (2), (4)$$

$$x_{i,j,s} = 0 \qquad \forall i, j \in T : i < j, \forall s \in S : h_i(s) = O \vee h_j(s) = O \vee h_i(s) = h_j(s) \quad (6)$$

The model is similar to formulation IP-GOP, with the difference that constraints (6) enforce the given HAP set. A HAP set is therefore feasible if and only if $z_H = \binom{n}{2}$. Again, the question arises whether more efficient conditions exist to detect infeasible HAP sets.

3.1. Complexity status of the HAP set feasibility problem

Let us first consider the time-relaxed bipartite HAP set feasibility problem (RBHAP) which asks whether a time-relaxed bipartite 1RR timetable compatible with a given pattern set exists. Lemma 4 states that RBHAP is $\mathcal{NP}$-complete. We use this result to prove Theorem 3 which states that the time-relaxed HAP set feasibility problem (RHAP) is $\mathcal{NP}$-complete.

Lemma 4. *The time-relaxed bipartite HAP set feasibility problem (RBHAP) is $\mathcal{NP}$-complete.*

Proof. We prove the theorem by presenting a reduction from completing sparsely-filled Latin squares (CLS) to RBHAP. Easton and Parker [19] prove that CLS is $\mathcal{NP}$-complete, even if only $3m$ cells are filled in any $m \times m$ Latin square.

CLS. *Instance*: A partial Latin square P, that is an $m \times m$ array filled with $3m$ symbols from $\psi = \{1, \ldots, m\}$ in such a way that each symbol occurs at most once in every row and column. *Question*: Is it possible to fill the empty cells in P so that every symbol occurs exactly once in every row and column?

For any instance of CLS, we create a set of m row teams T_1 and m column teams T_2 representing each row and column in P. In addition, we partition the set of time slots into two subsets $S = S_1 \cup S_2$. The first set contains a time slot for each symbol in CLS ($S_1 = \psi$), the second set contains a time slot $s_{i,j}$ for each filled cell $P_{i,j}$. The home-away pattern of a row team $i \in T_1$ (column-team $j \in T_2$) is such that this team has an off day during time slots in S_1 corresponding to symbols that are already present in row i (column j) of P. Moreover, team $k \in T$ has an off day during all time slots $s_{i,j} \in S_2$ with $k \neq i$ and $k \neq j$. In all other time slots, teams in T_1 have an 'H' in their pattern, whereas teams in T_2 have an 'A' in their pattern. We note that this transformation can be realized in polynomial time since the number of fixed cells is $3m$, resulting in $|S_1| + |S_2| = 4m$ time slots.

Now, we show that the CLS instance is feasible if and only if the corresponding RBHAP instance is feasible. Suppose first we have a feasible solution for the CLS instance, that is a Latin square L that fills every empty cell of P. Then, we can construct a feasible timetable for the RBHAP instance by scheduling game $(i, j), i \in T_1, j \in T_2$ on time slot $L_{i,j}$ whenever $P_{i,j}$ is empty, or on time slot $s_{i,j} \in S_2$ when $P_{i,j}$ is filled. This makes that all games are scheduled. In addition, each team plays at most one game per time slot in S_1 since each symbol in a Latin square occurs at most once in every row and column. The O's in the home-away patterns on time slots in S_1 are also respected since we did not schedule any game involving team $i \in T_1$ ($j \in T_2$) on time slots corresponding to filled symbols in row i (column j) of P. Similarly, the 'H's and 'A's in the home-away patterns on time slots in S_1 are respected since L completes P, and the home-away status of games is such that teams in T_1 (T_2) always play home (away).

Conversely, assume that we have a feasible timetable for the RBHAP instance. Then, exactly one game between teams $i \in T_1$ and $j \in T_2$ must be scheduled in S_1 whenever cell $P_{i,j}$ is empty. Indeed, it follows from the construction of the instance that the patterns of team i and j never simultaneously contain an 'H' and an 'A' during time slots in S_2. Therefore, we can always set the value of an empty cell $P_{i,j}$ to the time slot on which i plays against j. This results in a feasible Latin square completing P since teams have an off day during time slots that already occur in the corresponding row or column of P and teams in the RBHAP instance can play at most once per time slot. □

Theorem 3. *The time-relaxed HAP feasibility problem (RHAP) is $\mathcal{NP}$-complete.*

Proof. We prove the theorem by presenting a reduction from RBHAP. Recall, in RBHAP we are given a set of teams $T' = T'_1 \cup T'_2$ with $|T'_1| = |T'_2|$, a set of time slots S', and a home-away pattern h'_i for each team $i \in T'$. RBHAP is proven to be $\mathcal{NP}$-complete in Lemma 4.

We construct an instance of RHAP from any instance of RBHAP as follows. First, we set $T = T'_1 \cup T'_2$ and $S = S_1 \cup S_2 \cup S_3$, with $S_1 = S'$ and $|S_2| = |S_3| = |T'_1| - 1$ if $|T'_1|$ is even and $|S_2| = |S_3| = |T'_1|$ otherwise. We then construct the HAP of a team in S_1 by copying the HAP of the corresponding team in the input of the RBHAP instance. Next, we use the circle method (see de Werra [20]) to construct a (time-constrained) 1RR between the teams in T'_1; the home-away status can be chosen arbitrarily. For teams in T'_1, the HAP for time slots in S_2 corresponds to the home advantage of the games it plays in the above constructed 1RR; teams in T'_2 always have an off day during time slots in S_2. To construct the HAPs in S_3, we repeat this procedure but invert the role of teams in T'_1 and T'_2.

We now show that the RBHAP instance is feasible if and only if the RHAP instance is feasible. When given a feasible solution for the RBHAP instance, we construct a timetable for the RHAP instance as follows. First, we copy all game to time slot assignments in S_1. This schedules all games involving a team in T'_1 and a team in T'_2, while respecting the HAPs since the patterns are the same in S_1. It follows from the feasibility of the RBHAP solution that each team plays at most one game during time slots in S_1. Next, we schedule all games involving two teams in T'_1 in S_2 by copying the game assignments from the 1RR we constructed to determine the HAPs in S_2. Teams in T'_2 always have an off day in S_2. Obviously, this makes that teams play at most one game per time slot in S_2 and that all home-away patterns are respected in S_2. We repeat this procedure to schedule all games between the teams in T'_2 on time slots in S_3. This makes that all games are scheduled, and hence the RHAP solution is feasible.

Conversely, assume that we are given a feasible solution for the RHAP instance. Then, it must be that all games involving a team in T'_1 and a team in T'_2 are scheduled in S_1 since teams in T'_1 always have an off day in S_3 and teams in T'_2 always have an off day in S_2. Since the HAPs in S_1 are equal in the two instances, we can thus copy all game to time slot assignments in S_1. It follows from the feasibility of the RHAP solution that all games are scheduled and that each team plays at most one game per time slot, and hence the RBHAP solution is feasible. □

3.2. Necessary conditions for HAP set feasibility

Theorem 3 states that the time-relaxed HAP set feasibility problem is $\mathcal{NP}$-complete. In this section, we analyze some nec-

essary conditions.

If we denote with $c_H(T', s)$ and $c_A(T', s)$ the number of 'H's and 'A's in the patterns of a subset $T' \subseteq T$ of teams on time slot s, then the number of games between teams in T' on s is at most $\min(c_H(T', s), c_A(T', s))$. Hence, for the answer to be 'yes', Condition 5 is a necessary condition for both the time-constrained and time-relaxed HAP set feasibility problem.

Condition 5 (Miyashiro et al. [9])**.** *For each $T' \subseteq T$, we have* $\sum_{s \in S} \min(c_H(T', s), c_A(T', s)) \geq \binom{|T'|}{2}$.

For time-constrained patterns with an even number of teams and the minimal number of breaks (i.e., successions of two 'H's or 'A's in the pattern of a team), Miyashiro et al. [9] show that Condition 5 can be verified in polynomial time and conjecture Condition 5 to be sufficient. However, sufficiency is not proven and, therefore, complexity of the HAP set feasibility problem is open so far for arbitrary HAP sets as well as for time-constrained HAP sets having minimum number of breaks.

Briskorn [10] derives Condition 6 based on the LP-relaxation of IP-HAP.

Condition 6 (Briskorn [10])**.** *Denote with $\bar{z}$ the objective value of the LP-relaxation of IP-HAP, then* $\bar{z}_{\text{H}} = \binom{n}{2}$.

Since $z_{\text{H}} \leq \bar{z}_{\text{H}} \leq \binom{n}{2}$ and since a given HAP set is feasible if and only if $z_{\text{H}} = \binom{n}{2}$, Condition 6 is a necessary condition for both time-constrained and time-relaxed HAP set feasibility, and can be checked in polynomial time.

When the number of teams is even and the timetable is time-constrained, Briskorn [10] proves that Condition 6 is strictly stronger than Condition 5. Moreover, Briskorn [10] conjectures that Condition 6 is sufficient and asks whether z_{H} always coincides with $\bar{z}_{\text{H}}$. Horbach [21] answers the latter question negatively by providing an example of a HAP set for which $z_{\text{H}} \neq \bar{z}_{\text{H}}$. However, as the example concerns an infeasible HAP set, the sufficiency conjecture of Briskorn [10] is, after more than 10 years, still open.

Kashiwabara [22] rewrites the conjecture of Briskorn [10] in terms of perfect matchings, and uses a computer-assisted proof to confirm Briskorn's conjecture for 6 teams. Moreover, Kashiwabara [22, Conjecture 21] generalizes Briskorn's conjecture to the time-constrained bipartite HAP set feasibility problem; unless $\mathcal{P} = \mathcal{NP}$, it follows from Lemma 4 that the generalized conjecture does not hold for the time-relaxed setting.

We use a constraint programming solver (CPLEX CP Optimizer) to enumerate all time-constrained HAP sets and we check whether Condition 6 is violated whenever $z_{\text{H}} < \binom{n}{2}$ (IP-HAP solved with CPLEX). Following this procedure, we are able to confirm sufficiency of Condition 6 for the time-constrained setting with $n \leq 8$ and n even.

Bao [5] later used Condition 6 to verify the feasibility of time-relaxed HAP sets but could not find any example showing that the condition is not sufficient.

Observation 1. *Whereas sufficiency of Condition 6 is conjectured for time-constrained HAP set feasibility with an even number of teams (see Briskorn [10]), Table 6 shows that Condition 6 is not sufficient for time-relaxed HAP set feasibility.*

Table 6: Example of an infeasible HAP set for $n = 6$, satisfying Condition 6 ($\bar{z}_{\text{H}} = 15$). In the LP-relaxation, we have $x_{1,2,1} = x_{1,4,1} = x_{2,3,1} = x_{4,5,1} = x_{5,6,1} = x_{6,3,1} = x_{1,4,2} = x_{3,6,2} = x_{4,2,2} = x_{5,2,2} = x_{5,3,2} = x_{6,1,2} = x_{1,3,3} = x_{1,6,3} = x_{2,5,3} = x_{3,2,3} = x_{4,5,3} = x_{4,6,3} = x_{2,1,4} = x_{3,1,4} = x_{4,2,4} = x_{4,6,4} = x_{5,3,4} = x_{5,6,4} = 0.5$ and $x_{1,5,5} = x_{2,6,5} = x_{3,4,6} = 1$.

T/S	1	2	3	4	5	6
1	A	H	H	H	A	O
2	H	H	H	A	H	O
3	A	H	A	A	O	H
4	H	A	H	H	O	A
5	A	A	A	H	H	O
6	H	A	A	A	A	O

Acknowledgements: The authors wish to thank Frits C.R. Spieksma and Roel Lambers for their constructive feedback.

References

[1] D. Van Bulck, D. Goossens, J. Schönberger, M. Guajardo, RobinX: A three-field classification and unified data format for round-robin sports timetabling, Eur. J. Oper. Res. 280 (2020) 568 – 580.
[2] J. Schönberger, D. C. Mattfeld, H. Kopfer, Memetic algorithm timetabling for non-commercial sport leagues, Eur. J. Oper. Res. 153 (2004) 102–116.
[3] D. Van Bulck, D. R. Goossens, F. C. R. Spieksma, Scheduling a non-professional indoor football league: a tabu search based approach, Ann. Oper. Res. 275 (2019) 715–730.
[4] D. Van Bulck, D. Goossens, Handling fairness issues in time-relaxed tournaments with availability constraints, Comput. Oper. Res. 115, (2020).
[5] R. Bao, Time relaxed round robin tournament and the NBA scheduling problem, Ph.D. thesis, Cleveland State University, 2009.
[6] G. L. Nemhauser, M. A. Trick, Scheduling a major college basketball conference, Oper. Res. 46 (1998) 1–8.
[7] K. Easton, Using integer programming and constraint programming to solve sports scheduling problems., Ph.D. thesis, Georgia Institute of Technology, USA, 2003.
[8] D. Briskorn, The Home-Away-Pattern Set Feasibility Problem, MAPSP, 2017.
[9] R. Miyashiro, H. Iwasaki, T. Matsui, Characterizing Feasible Pattern Sets with a Minimum Number of Breaks, in: E. Burke, P. De Causmaecker (Eds.), Practice and Theory of Automated Timetabling IV, Springer Berlin Heidelberg, Berlin, Heidelberg, 78–99, 2003.
[10] D. Briskorn, Feasibility of home-away-pattern sets for round robin tournaments, Oper. Res. Lett. 36 (2008) 283 – 284.
[11] D. Goossens, F. Spieksma, Breaks, cuts, and patterns, Oper. Res. Lett. 39 (2011) 428 – 432.
[12] M. Davari, D. Goossens, J. Beliën, R. Lambers, F. Spieksma, The multi-league sports scheduling problem, or how to schedule thousands of matches, Oper. Res. Lett. 48 (2020) 180 – 187.
[13] U. Schauz, The tournament scheduling problem with absences, Eur. J. Oper. Res. 254 (2016) 746 – 754.
[14] D. Briskorn, A. Drexl, F. C. R. Spieksma, Round robin tournaments and three index assignments, 4OR 8 (2010) 365–374.
[15] R. Burkard, M. Dell'Amico, S. Martello, Assignment problems, Springer, 2009.
[16] G. J. Chang, Complete Diagonals of Latin Squares, Can. Math. Bulletin 22 (1979) 477–481.
[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2009.
[18] D. Bryant, C. A. Rodger, On the completion of Latin rectangles to symmetric Latin squares, J. Aust. Math. Soc. 76 (1) (2004) 109–124.
[19] T. Easton, R. G. Parker, On completing Latin squares, Discrete Appl. Math. 113 (2001) 167 – 181.
[20] D. de Werra, Scheduling in sports, in: P. Hansen (Ed.), Studies on graphs and discrete programming, North-Holland, Amsterdam, 381–395, 1981.
[21] A. Horbach, A combinatorial property of the maximum round robin tournament problem, Oper. Res. Lett. 38 (2010) 121 – 122.
[22] K. Kashiwabara, Scheduling partial round robin tournaments subject to home away pattern sets, The Electronic Journal of Combinatorics 16 (2009) R55.