Breaks, cuts, and patterns

Dries R. Goossens[a,b,c], Frits C.R. Spieksma[b,c]

[a] *PostDoc researcher for Research Foundation - Flanders*
[b] *Center for Operations Research and Business Statistics (ORSTAT)*
[c] *Faculty of Business and Economics, K.U.Leuven, Belgium*

Abstract

We generalize the concept of a break by considering pairs of arbitrary rounds. We show that a set of home-away patterns minimizing the number of generalized breaks cannot be found in polynomial time, unless P=NP. When all teams have the same break set, the decision version becomes easy; optimizing remains NP-hard.

Keywords: sport scheduling, home-away patterns, breaks, nonconsecutive rounds, complexity

1. Introduction

Consider a sports competition in which an even number of teams (say $2n$) compete. Each team has its own venue, and each pair of teams meets once in one of the teams venues. Clearly, this is a single round robin tournament which can be played in $2n-1$ rounds. A round is a set of games, usually played simultaneously, in which every team plays at most one game. A schedule is called compact when it uses the minimum number of rounds required to schedule all the games. In a compact schedule with an even number of teams, each team plays exactly one game in each round. If the league contains an odd number of teams, a dummy team may be added, reducing this situation to the case with an even number of teams. In each round, the team playing against the dummy team has a 'bye', i.e. does not play. Adding a dummy team, however, allows to create a schedule without breaks, even when we consider breaks around byes (see Froncek and Meszka [1]). In this work, we deal solely with compact schedules for an even number of teams.

Traditional terminology prescribes that the sequence of home matches ('H') and away matches ('A') played by a single team is called its *home-away*

pattern (HAP). Given such a HAP, the occurrence of two consecutive home matches, or two consecutive away matches is called a *break*.

In this contribution, we will generalize the concept of a break. The idea is simple: instead of defining a break as two home games (or two away games) in a pair of consecutive rounds, we will view a break as two home (away) games in a given, pair of *arbitrary* rounds. More specific, each team i, $i = 1, \ldots, 2n$ specifies a set of pairs of rounds indicating that this team does not want to play either at home or away in both rounds of each pair (which need not be consecutive). The set of pairs is called the *break set* of team i, and is denoted by B_i, with $i = 1, \ldots, 2n$. It generalizes the traditional concept of a break. Indeed, the traditional setting arises when:

$$B_1 = B_2 = \ldots = B_{2n} = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \ldots, \{2n - 2, 2n - 1\}\}.$$

We say that a home-away pattern (HAP) is *break-free* with respect to a break set B_i if no two home games or two away games are planned on a pair of rounds that is an element of B_i. We call a set of $2n$ home-away patterns a *pattern set* if they are pairwise distinct, and when each round has n H's (and hence n A's). We call a pattern set *break-free* if the i-th HAP in the set is break-free with respect to B_i. The main problem can now be described as follows: given $2n$ break sets B_i, with $i = 1, \ldots, 2n$, does there exist a break-free pattern set with respect to B_i? We refer to this problem as the BfPS problem.

In this paper, we present the following results. We show that when given a break set B_i for each team i, $i = 1, \ldots, 2n$, deciding whether a break-free pattern set exists is NP-complete (Section 4). In the special case where all teams have identical break sets, we show that the BfPS problem can be solved in polynomial time, while minimizing the number of generalized breaks remains NP-hard (Section 5). For this special case, we also provide a lower bound for the number of breaks, generalizing a classical result by De Werra [2]. In the next section, we motivate the concept of generalized breaks; an overview of related problems and results is provided in Section 3.

2. Motivation

In most major sport leagues, a distinction is made between matches depending on the day on which they are scheduled. For instance, in the 1992

Cricket World Cup, only matches on Wednesday, Saturday and Sunday could be broadcasted on TV (Armstrong and Willis [3]). When scheduling a major college basketball league, Nemhauser and Trick [4] take into account the fact that teams value weekend home games higher than midweek home games. In football leagues, the vast majority of the matches are scheduled on weekend days. However, as there are often more rounds than available weekends, some rounds need to be scheduled on Wednesdays. Since a home game on a Wednesday typically attracts less fans (for instance because of overlap with the Champions League, see Forrest and Simmons [5]), teams generally do not appreciate a home game on a midweek round. Consequently, teams ask for a schedule where the assignment of home games on Wednesdays is balanced: if a team plays at home on one Wednesday round, they don't want to play at home on the next Wednesday round. This is for instance the case in the Belgian Jupiler Pro Football League (Goossens and Spieksma [6]), and the Chilean First Division (Durán et al. [7]). Obviously, Wednesday rounds need not be consecutive, and hence generalized breaks arise.

When confronted with generalized breaks, solving the BfPS problem produces relevant HAP sets, which may be the first step in a scheduling procedure. Indeed, this approach can easily be plugged in to the popular "first break then schedule" approach, where first the home away patterns are determined and assigned to teams, and afterwards the actual matches are fixed (see e.g. Easton et al. [8]). Although a BfPS solution satisfies, by definition, some necessary conditions, it is not guaranteed that there exists a corresponding round robin tournament. It remains an open question whether this can be checked in polynomial time. Briskorn [9], however, provides an IP formulation that can answer this question efficiently for most practical values of $2n$.

3. Related work

Kendall et al. [10] present an annotated bibliography of sports scheduling literature. Many of the theoretical results and scheduling algorithms in this bibliography are based on graph theory. For an overview of graph-based models in sports scheduling, we refer to the work by Drexl and Knust [11]. These authors mention that, to the best of their knowledge, De Werra [2] was the first who suggested to use graphs for constructing schedules with home and away games. De Werra [2] uses the complete graph K_{2n} on $2n$ nodes for

single round robin tournaments, where the nodes correspond with the teams, and the edges with games between the teams. A compact schedule can then be seen as an edge coloring with $2n-1$ colors, i.e. a partitioning of the edge set into $2n-1$ perfect matchings.

Apart from introducing graph theory in sports scheduling, De Werra [2] also considers the problem of finding the minimum number of breaks (in the classic interpretation, i.e. with consecutive rounds) for a single round robin tournament with $2n$ teams. He finds that since only two different patterns without breaks exist (HAHA...H and AHAH...A), and all teams must have different patterns (indeed, two teams with the same pattern can never play against each other), at most two teams will not have any break. Consequently, in each schedule at least $2n-2$ breaks occur. Furthermore, De Werra shows that schedules achieving this lower bound can be constructed using the so-called canonical 1-factorization, which dates back to, as far as we are aware, a paper by Kirkman [12]. In this work, we partly extend De Werra's result to our setting with generalized breaks.

If predetermined, a set of home-away patterns restricts the set of possible matches. Of course, two teams should have a different home-away assignment in some round in order to be able to play against each other in that round. Obviously, there is no guarantee that a random set of HAPs allows a feasible single round robin schedule (i.e. where each team plays exactly once against each other team). The question whether or not it is possible to create a feasible single round robin schedule with a given set of home-away patterns is known as the *pattern set feasibility problem*. The complexity of this problem is still unsettled, but a number of necessary conditions for the answer to be 'yes' are known.

Miyashiro et al. [13] point out that it is easy to see that every feasible set of HAPs must satisfy the following two conditions:

1. in each round, the number of As and Hs are equal,
2. the HAPs are pairwise different.

Notice that these conditions correspond with our definition of a *pattern set* in section 1. They are, however, not sufficient. In other words, finding a break-free pattern set does not guarantee the existence of a schedule without

(generalized) breaks.

Miyashiro et al. [13] also come up with more elaborate necessary conditions, which can be checked in polynomial time for sets of patterns with a minimum number of breaks. For this case, the authors checked computationally that the conditions are sufficient for problems with up to 26 teams. Briskorn [9] provides a necessary condition based on a linear programming formulation. He shows that this condition is strictly stronger than those provided by Miyashiro et al. [13], but it remains an open question whether it is sufficient (see also Horbach [14]).

Another related problem is the break minimization problem: given a tournament schedule without home-away assignment, find a feasible home-away pattern set that minimizes the number of (classic) breaks. To the best of our knowledge, the complexity status of this problem is still unknown. However, Miyashiro and Matsui [15] proved a conjecture by Elf et al. [16], stating that deciding whether a feasible home-away pattern set with $2n-2$ breaks exists, can be solved in polynomial time.

4. The break-free pattern set problem

We prove that the BfPS problem is NP-complete using a reduction from a problem we call the *matrix flopping problem* (MFP). In the matrix flopping problem, we are given a 0-1 matrix M, consisting of $2k$ rows and $2k-1$ columns. Moreover, each row contains at least one '1' and one '0', and all rows are pairwise distinct and noncomplementary. A *flop* is defined as changing each '1' into a '0' and vice versa, for all the entries on some row i, with $1 \leqslant i \leqslant 2k$ (we will refer to this operation as *flopping* a row). The question is whether there exists a subset of rows to be flopped such that as a result, each column sum equals k.

Lemma 4.1. *The matrix flopping problem is NP-complete.*

Proof. We prove the theorem by presenting a reduction from a special case of Exact Cover by 3-sets (X3C), namely where each element occurs in exactly three subsets. We refer to this problem as X3C3. Garey and Johnson [17] showed that X3C is NP-complete. Hein et al. [18] implied that X3C remains

NP-complete when each element occurs in exactly three subsets. The NP-completeness of X3C3 was explicitly proven by Hickey et al. [19].

X3C3
Input: A set of elements X with $|X| = 3q$, and a collection C of $m = 3q$ 3-element subsets of X, such that each element occurs in exactly 3 members of C.
Question: Does C contain an exact cover for X, i.e. does there exist a sub-collection $C' \subset C$ such that every element of X occurs in exactly one member of C'?

Each instance of X3C3 can be transformed into an instance of MFP as follows. We construct a matrix M with $6q-2$ rows and $6q-3$ columns. We call the first $3q$ columns the *left part* of the matrix, the next column is called the *target* column, and the last next $3q-4$ columns are the *right part*. The first $3q$ rows are denoted as the *upper part* of the matrix, the next $3q-6$ rows are the *middle part*, and the last 4 rows form the *lower part*. The left part columns correspond with the $3q$ elements in X, the upper part rows correspond with the $3q$ members of C. Each entry m_{ij} in the upper left part of the matrix has value 1 if the j-th element of X occurs in the i-th member of C, and 0 otherwise. Thus, the left upper part of M captures the instance of X3C3. The entries in the middle left part of the matrix all have value 1. In the lower left part, all entries in the second and third row are 1, the entries in the first and last row have value 0. The target column has in its upper part all zeros, except on the three rows corresponding to the 3 members of C in which the first element of X occurs. The middle part of the target column consists of $q-2$ ones, followed by $2q-4$ zeros. The lower part of the target column has a zero in the first and the third row, and a one in the second and the last row. In the upper part of the matrix, the right part columns are the complement of the target column. In the middle part, the entries in the first $q-2$ rows have value 1, except that $m_{ij} = 0$ for $i = 3q+1, ..., 4q-2$ with $j = i+1$. The next $q-2$ rows have value 0, except that $m_{ij} = 1$ for $i = 4q-1, ..., 5q-4$ with $j = i-q+3$. The last $q-2$ middle part rows again have value 1, except that $m_{ij} = 0$ for $i = 5q-3, ..., 6q-7$ with $j = i-2q+5$ and $i = 5q-2, ..., 6q-6$ with $j = i-2q+4$. In the first two rows of the lower part, all entries have value 1 in columns $3q+2$ to $4q-2$, and 0 in the next columns. The last two rows of the lower part consist of zeros in the right part. We illustrate this transformation with the following matrix M,

where lines have been inserted to show the various parts. The target column is indicated in boldface.

$$\mathbf{M} = \left(\begin{array}{cccccc|c|cccccccc}
0 & 0 & 0 & 1 & \dots & 0 & \mathbf{0} & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
0 & 1 & 0 & 0 & \dots & 0 & \mathbf{0} & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
1 & 1 & 0 & 0 & \dots & 0 & \mathbf{1} & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 1 & 1 & 1 & \dots & 0 & \mathbf{0} & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
0 & 0 & 0 & 0 & \dots & 1 & \mathbf{0} & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
1 & 0 & 1 & 0 & \dots & 0 & \mathbf{1} & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
1 & 0 & 0 & 0 & \dots & 0 & \mathbf{1} & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 1 & 0 & \dots & 1 & \mathbf{0} & 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
\hline
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{1} & 0 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{1} & 1 & 0 & \dots & 1 & 1 & 1 & \dots & 1 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{1} & 1 & 1 & \dots & 0 & 1 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{1} & 1 & 1 & \dots & 1 & 0 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 1 & \dots & 0 & 0 & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 0 & \dots & 1 & 0 & 0 & \dots & 0 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 0 & \dots & 1 & 1 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 1 & 0 & \dots & 1 & 1 & 1 & \dots & 1 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 1 & 1 & \dots & 0 & 1 & 1 & \dots & 1 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 1 & 1 & \dots & 0 & 1 & 1 & \dots & 1 \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & \mathbf{0} & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{1} & 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\
1 & 1 & 1 & 1 & \dots & 1 & \mathbf{0} & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & 0 & \dots & 0 & \mathbf{1} & 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0
\end{array}\right)$$

Observe that the rows of M are indeed pairwise distinct and noncomplementary. Notice also that for each column in the left part, the sum over the

rows equals $3q-1$, i.e. exactly half of the number of rows. For the target column, however, the sum equals $q+3$, and for the right part columns $5q-7$. This completes the description of the instance of MFP.

We now show that a yes-answer for the X3C3 instance corresponds to a feasible solution for the MFP instance. If X3C3 has a solution, this means that there exist q rows from the upper part of M such that for each column in the left part, the sum over these rows is exactly 1. Flopping these rows will switch in each left part column $q-1$ zeros into ones, and 1 one into a zero. In other words, the column sums for the left part now all increased by $q-2$. To compensate for this, we also flop the last $q-2$ rows of the middle part, and the first 2 rows of the lower part. Observe that the upper parts of the target column and the first column are identical. Consequently, in the upper part of the target column, $q-1$ zeros are switched to one, and 1 one was switched to zero. In the middle part of the target column, flopping the last $q-2$ rows results in as many extra 1-entries. In total, this brings the sum over all the rows for the target column to $3q-1$. In the right part columns, the flops cause a column sum to decrease by $q-2$ in the upper part rows, and again by $q-4$ in the other flopped rows, such that these column sums now also equal $3q-1$. Thus, the X3C3 solution determines a set of row flops, such that the sum over all rows equals $3q-1$ for each column of M.

A yes-answer to the MFP instance in turn corresponds to a yes-answer of the X3C3 instance. Let's determine how this solution for MFP looks like. We argue that if a solution for MFP exists, it must flop exactly q or $2q$ rows from the upper part. Notice that for the left part of M, the column sums already equal $3q-1$. This is not the case for the target column, for which the sum is $q+3$. Therefore, a solution will always require at least one flop in the upper part. Indeed, flopping rows from the lower part could set the sum in the target column to the required $3q-1$, but will decrease the sum for each left part column, which can only be compensated by flopping upper part rows. Flopping just one upper part row will also never result in a solution, since this would result in 3 columns in the left part, where the sum over the rows is two lower than for the other left part columns. In fact, equal column sums in the left part can only be maintained if a multiple of q rows are flopped in the upper part. However, flopping all $3q$ upper part rows can never lead to a solution for our MFP problem. Indeed, it would increase the sum in each left part column by $3q-6$, which can only be compensated by

flopping at least $3q - 6$ middle or lower part rows. This would however set the target column sum to at least $5q + 1$. In conclusion, a solution for MFP flops exactly $2q$ or q rows from the upper part of the matrix. Moreover, in order to ensure that all columns in the left part have an equal sum, the q (or $2q$) rows must be flopped such that in each column precisely 1 (or 2) one(s) is (are) switched to zeros, and hence $q - 1$ (or $2q - 2$) zeros are switched to ones. This solution for MFP translates to a solution for X3C3 by selecting those members of C that correspond to the q rows in M from the upper part that were flopped (or the q rows that were not). □

Notice that both options in this proof in fact correspond to the same solution for X3C3. Notice also that if MFP has a solution, then its complement will also be a solution for MFP. In other words, given a MFP solution, flopping all rows will result in another MFP solution.

We now show that deciding whether a break-free pattern set exists is NP-complete, even if at most two home-away patterns are break-free with respect to B_i, for each team i. If no break-free home-away pattern exists for the break set of some team i, then obviously no break free pattern set exists. Checking whether a break set B_i has a break-free pattern can be done in polynomial time (see section 5). Therefore, without loss of generality, we assume for the remainder of this section that each break set B_i allows at least one break-free home-away pattern.

Theorem 4.2. *Given a break set B_i for each team $i = 1, \ldots, 2n$, deciding whether a break-free pattern set exists is NP-complete, even if at most two home-away patterns are break-free with respect to B_i, for each team i.*

Proof. Each instance of MFP can be transformed into an instance of the BfPS problem as follows. Given a matrix M with $2k$ rows and $2k - 1$ columns, each row corresponds with a team, and each column with a round. Each row i in M contains at least one '1' and one '0'. We can construct a break set B_i with $i = 1, ..., 2k$ by adding to B_i a pair of rounds for each pair of columns for which different entries can be found in row i. Hence, each B_i will allow at most two break-free home-away patterns.

A yes-answer for the MFP instance corresponds to a feasible solution for the BfPS instance. Indeed, after the row flops resulting in a MFP solution,

translating the ones into home games and the zeros into away games (or vice versa) produces a home-away pattern that is break-free with respect to B_i. For each round, there will be exactly as many home games as away games, and since the rows of M are pairwise distinct and noncomplementary, each team receives a different home-away pattern. A yes-answer to the BfPS instance in turn corresponds to a yes-answer of the MFP instance. If there exists a break-free pattern set, this means that, using the same translation as before, each row in the matrix can be flopped into the position corresponding to the HAP in the break-free pattern set, such that there are exactly k ones and k zeros in each column.

□

5. A special case: identical break sets

In this section we deal with the complexity of the problem in the special case where the team's break sets are identical, i.e., the case when $B \equiv B_1 = B_2 = \ldots = B_{2n}$. We first consider the case of deciding whether there exists a break-free pattern set, and then we discuss the optimization variant where we want to find a pattern set minimizing the number of breaks.

5.1. Deciding

We will reformulate the problem by building a graph, and write our problem as a coloring problem. Construct a graph G_B as follows: there is a node for each round, and two nodes are connected if and only if the corresponding pair of rounds is in the set B. A coloring of G_B is an assignment of one out of two colors to each node. A coloring is called feasible when connected nodes receive different colors. Notice that a HAP for a specific team is a coloring of G_B and vice versa. Observe also that the existence of a feasible coloring means that G_B is bipartite. Let k_B be the number of connected components of G_B; in the sequel we will omit the subscript 'B', and simply use k instead. We will also use the symbol g_B, or g for short, denoting the number of bipartite connected components of G_B.

The decision variant of our problem arises when we ask for the existence of a pattern set with no breaks: given B, does there exist a break-free pattern set? This question can be answered in polynomial time.

Theorem 5.1. *Given B, the question whether or not a break-free pattern set exists, can be answered in polynomial time.*

Proof. We use the following procedure. Decide whether G_B is bipartite. If so, count the number of connected components k, and if $k \geq \log_2(2n)$, the answer is yes; otherwise the answer is no. This procedure is correct, since

1. if G_B contains a component that is not bipartite, no feasible coloring of that component, and hence no break-free pattern set, exists,
2. in case each component is bipartite, there are exactly two feasible colorings of that component (by interchanging both colors).

Thus, given k components, we can exhibit 2^k pairwise distinct colorings such that each node is colored with one color 2^{k-1} times. If $2^k > 2n$, we simply remove $2^k - 2n$ colorings that form $\frac{2^k-2n}{2}$ complementary pairs. □

5.2. Optimizing

There is an intimate relation between our problem, and MAX-CUT. Indeed, recall that the (unweighted) MAX-CUT problem is to partition the vertex set of a given graph G into two sets such that the number of edges between vertices that are in different sets, is maximized. Notice that the graph G_B that results from a BfPS problem can be any graph. A MAX-CUT solution for G_B (i.e., a partition of V_B) can be translated to two complementary HAPs, by assigning a home game to rounds corresponding to the vertices in the first set, and an away game to those in the other set (and vice versa). Clearly, solving a MAX-CUT problem on G_B gives us a pair of complementary HAPs minimizing the number of breaks. Finding the best n MAX-CUT solutions on G_B gives us a pattern set with a minimum number of (generalized) breaks. Indeed, the resulting HAPs will be non-identical, and since they are generated in complementary pairs, the number of home games equals the number of away games for each round. Since an optimal pattern set, i.e., a pattern set with a minimum number of breaks, will always contain a pair of HAPs corresponding to an optimal MAX-CUT solution, solving the optimization version of the BfPS problem is at least as hard as solving the unweighted MAX-CUT problem on G_B. Given the above discussion, the following result will come as no surprise.

Theorem 5.2. *Given B, finding a pattern set that minimizes the number of breaks is NP-hard.*

Furthermore, given a graph $G_B = (V_B, E_B)$, if $|E_B| - l$ is an upper bound for MAX-CUT on G_B, then $2nl$ is a lower bound for the minimum number of breaks. Let us derive a lower bound for the number of breaks in any schedule given a break set B.

Theorem 5.3. *Given B, the number of breaks in any feasible schedule is at least $2n(k-g) + max(0, 2n - 2^g)$.*

Proof. Each component that does not admit a feasible coloring gives rise to at least one break in each of the $2n$ colorings: $2n(k-g)$. Each component that admits a feasible coloring, admits exactly two feasible colorings, and hence no more than 2^g colorings are break-free. □

Observe that in the traditional setting, where G_B is a path on $2n-1$ nodes, the lower bound coincides with De Werra's bound of $2n-2$. Also, observe that in case G_B is a tree $2n-2$ remains a valid lower bound.

Acknowledgement

We wish to thank two anonymous referees for their valuable comments on a previous draft of this paper.

References

[1] D. Froncek, M. Meszka, Round robin tournaments with one bye and no breaks in home-away patterns are unique, in: In G. Kendall, E. Burke, S. Petrovic, and M. Gendreau, editors, Multidisciplinary Scheduling: Theory and Applications, 2005, pp. 331-340. Springer, Berlin.

[2] D. De Werra, Scheduling in sports, in: P. Hansen (Ed.), Studies on Graphs and Discrete Programming, volume 11 of *Annals of Discrete Mathematics*, North-Holland, Amsterdam, 1981, pp. 381–395.

[3] J. Armstrong, R. Willis, Scheduling the cricket world cup - a case study, Journal of the Operational Research Society 44 (1993) 1067–1072.

[4] G. Nemhauser, M. Trick, Scheduling a major college basketball conference, Operations Research 46 (1998) 1–8.

[5] D. Forrest, R. Simmons, New issues in attendance demand: The case of the English football league, Journal of Sports Economics 7 (2006) 247–266.

[6] D. Goossens, F. Spieksma, Scheduling the Belgian soccer league, Interfaces 39 (2009) 109–118.

[7] G. Durán, M. Guajardo, J. Miranda, Suaré, S. Souyris, A. Weintraub, R. Wolf, Scheduling the Chilean Soccer League by integer programming, Interfaces 37 (2007) 539–552.

[8] K. Easton, G. Nemhauser, M. Trick, Sports scheduling, in: J. Leung (Ed.), Handbook of Scheduling, CRC Press, 2004, pp. 52.1–52.19.

[9] D. Briskorn, Feasibility of homeaway pattern sets for round robin tournaments, Operations Research Letters 36 (2008) 283–284.

[10] G. Kendall, S. Knust, C. Ribeiro, S. Urrutia, Scheduling in sports: An annotated bibliography, Computers and Operations Research 37 (2010) 1–19.

[11] A. Drexl, S. Knust, Sports league scheduling: Graph- and resource-based models, Omega 35 (2007) 465–471.

[12] T. Kirkman, On a problem in combinations, Cambridge and Dublin Math. J. 2 (1847) 191–204.

[13] R. Miyashiro, H. Iwasaki, T. Matsui, Characterizing feasible pattern sets with a minimum number of breaks, Lecture Notes in Computer Science 2740 (2003) 78–99.

[14] A. Horbach, A combinatorial property of the maximum round robin tournament problem, Operations Research Letters 38 (2010) 121–122.

[15] R. Miyashiro, T. Matsui, A polynomial-time algorithm to find an equitable home-away assignment, Operations Research Letters 33 (2005) 235–241.

[16] M. Elf, M. Jünger, G. Rinaldi, Minimizing breaks by maximizing cuts, Operations Research Letters 31 (2003) 343–349.

[17] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness., New York: W.H. Freeman and Co., 1979.

[18] J. Hein, T. Jiang, L. Wang, K. Zhang, On the complexity of comparing evolutionary trees, Discrete Applied Mathematics 71 (1996) 153–169.

[19] G. Hickey, F. Dehne, A. Rau-Chaplin, C. Blouin, SPR distance computation for unrooted trees, Evolutionary Bioinformics Online 4 (2008) 17–27.