ECONSTOR
Make Your Publications Visible.

A Service of
ZBW
Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Briskorn, Dirk; Drexl, Andreas

Working Paper — Digitized Version

A branching scheme based on First-Break-Then-Schedule decomposition

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 623

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Suggested Citation: Briskorn, Dirk; Drexl, Andreas (2007) : A branching scheme based on First-Break-Then-Schedule decomposition, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 623, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/147676

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 623

A Branching Scheme Based on First–Break–Then–Schedule Decomposition

Dirk Briskorn, Andreas Drexl

March 2007

Dirk Briskorn, Andreas Drexl
Christian-Albrechts-Universität zu Kiel,
Institut für Betriebswirtschaftslehre,
Olshausenstr. 40, 24098 Kiel, Germany,
http://www.bwl.uni-kiel.de/bwlinstitute/Prod
briskorn@bwl.uni-kiel.de, andreas.drexl@bwl.uni-kiel.de

Abstract

Single round robin tournaments are a well known class of sports leagues schedules. We consider leagues with a set T of n teams where n is even. Costs are associated to each possible match. The goal is to find the minimum cost tournament among those having the minimum number of breaks. We pick up structural properties of home-away-pattern sets having the minimum number of breaks. An branching idea using these properties is developed in order to guide branching steps on the first levels of a branch-and-bound tree in order to avoid nodes corresponding to infeasible subproblems.

Keywords: Sports league scheduling, round robin tournaments, home-away-pattern set, break, branch-and-bound

1 Introduction

Round robin tournaments (RRT) cover a huge variety of different types of sports league schedules arising in practice. A set of teams T, $|T| = n$ even, competes such that each team plays exactly once against each other team, either at home or away. Furthermore, a team $i \in T$ has to play exactly once in each period and, hence, we have a set P of $n-1$ periods altogether. Therefore, a set of periods P, $|P| = n-1$, is given.

An illustrative example for a single RRT where $n = 6$ is given in table 1. Here i-j denotes that team i plays at home against team j.

	1	2	3	4	5
match 1	3-4	4-5	2-4	4-6	6-5
match 2	5-2	1-3	5-1	3-5	4-1
match 3	6-1	2-6	6-3	1-2	2-3

Table 1: Single RRT for $n = 6$

Since each match has to be carried out at one of the venues of both opponents breaks come into play. We say team i has a break in period p if and only if i plays twice at home or away, respectively, in periods $p-1$ and p. For the sake of fairness among teams the most common goal concerning breaks is to minimize the number of their occurences. It is well known from de Werra [9] that the number of breaks can not be less than $n-2$ in a single RRT. The single RRT provided in table 1 has 4 breaks for 6 teams and, therefore, has the minimum number of breaks.

Models for sports league scheduling have been the topic of extensive research. A whole stream of papers is based on the analogy between sports league scheduling and edge coloring of complete graphs. Examples are de Werra [9, 10, 11, 12, 13], de Werra et al. [14], and Drexl and Knust [15]. Brucker and Knust [8] and Drexl and Knust [15] analyze the relationship between sports league scheduling and multi-mode resource constrained project scheduling. Briskorn et al. [7] line out the similarity of structures of single RRTs and planar three index assignments. Bartsch [2], Bartsch et al. [3], and Schreuder [19, 20] examine particular formulations.

A common approach in order to schedule a sports league is to decompose the decision problem into two subproblems to be solved sequentially. First, the venue (home/away) for each team

is fixed in each period. The result can be represented as a home-away-pattern (HAP) set $h \in \{0,1\}^{n \times n-1}$. The entry of $h_{i,p}$ corresponding to line i and column p equals 0 or 1 if team i is fixed to play at home or away, respectively, in period p. The second step is to schedule matches according to the given HAP set h which means a match between teams i and j in period p is possible if and only if $h_{i,p} \neq h_{j,p}$. Then, this match is carried out at team i's home if and only if $h_{i,p} = 0$. Obviously, an arbitrary HAP set h does not necessarily allow a single RRT to be arranged. In the remainder a HAP set h is called feasible if a single RRT can be arranged based on h. There is no sufficient condition known for a HAP set to be feasible which can be checked in polynomial time. However, Miyashiro et al. [16] provide a strong necessary condition for HAP sets with the minimum number of breaks which can be checked in polynomial time. Briskorn [4] propose a necessary condition for general HAP sets and proof it to be stronger than the one in Miyashiro et al. [16]. Moreover, the condition can be checked in polynomial time and is conjectured to be sufficient.

The remainder of this paper is organized as follows. In section 2 we define a sports league scheduling problem and represent it by means of IP models. In section 3 we develop a guiding scheme to be employed in a branching framework in order to solve the sports league scheduling problem. In section 4 we line out computational results and, finally, section 5 contains conclusions.

2 Problem Definition and Model

We associate cost $c_{i,j,p}$ to the match between teams i und j in period p at i's home. Cost $c_{i,j,p}$ of a specific match can be seen in a rather abstract way here. However, there are several application of $c_{i,j,p}$ with practical relevance, see Briskorn et al. [7]. We define the cost of a tournament as the sum of arranged matches' cost and consider the problem to find the minimum cost single RRT having the minimum number of breaks. This problem can be represented by an IP model as introduced in Briskorn and Drexl [6].

We use binary variable $x_{i,j,p}$, $i \in T$, $j \in T$, $j \neq i$, $p \in P$, equaling 1 if and only if team i plays at home against team j in period p. Then, objective function (1) represents the goal of cost minimization. Constraints (2) and (3) assure the single RRT structure. Restriction (2) forces each pair of teams to meet exactly once. Constraint (3) ensures that each team plays exactly once per period. In order to consider breaks we employ binary variables $br_{i,p}$, $i \in T$, $p \in P^{\geq 2}$, being equal to 1 if and only if team i has a break in period p. Then, constraints (4) or (5) force $br_{i,p}$ to equal 1 if team i plays twice at home or twice away, respectively, in periods $p-1$ and p. Restriction (6) limits the number of breaks to the minimum number. Note that combining (4), (5), and (6) leads to $br_{i,p}$, $i \in T$, $p \in P^{\geq 2}$, equaling 1 if and only if i has a break in p.

3 Branch–and–Bound Algorithm

This section focuses on a branching scheme in line with decomposition schemes following the first-break-then-schedule idea as sketched in section 1. We consider the LP relaxation of the problem at hand, solve it to optimality, and branch on the venue of team i in period p. Accordingly, we construct a branching tree where a path from the root node to a leaf either can be pruned or represents a HAP set.

$$\min \sum_{i \in T} \sum_{j \in T \setminus \{i\}} \sum_{p \in P} c_{i,j,p} x_{i,j,p} \tag{1}$$

$$\begin{aligned}
\text{s.t.} \sum_{p \in P} (x_{i,j,p} + x_{j,i,p}) &= 1 && \forall\, i,j \in T, i < j && (2)\\
\sum_{j \in T \setminus \{i\}} (x_{i,j,p} + x_{j,i,p}) &= 1 && \forall\, i \in T, p \in P && (3)\\
\sum_{j \in T \setminus \{i\}} (x_{i,j,p-1} + x_{i,j,p}) - br_{i,p} &\leq 1 && \forall\, i \in T, p \in P^{\geq 2} && (4)\\
\sum_{j \in T \setminus \{i\}} (x_{j,i,p-1} + x_{j,i,p}) - br_{i,p} &\leq 1 && \forall\, i \in T, p \in P^{\geq 2} && (5)\\
\sum_{i \in T} \sum_{p \in P^{\geq 2}} br_{i,p} &\leq n-2 && && (6)\\
x_{i,j,p} &\in \{0,1\} && \forall\, i,j \in T, i \neq j, p \in P && (7)\\
br_{i,p} &\in \{0,1\} && \forall\, i \in T, p \in P^{\geq 2} && (8)
\end{aligned}$$

3.1 Basic Idea

Branching on a subset of variables to have values' sum in $\{0,1\}$, respectively, is a well known generalization of $\{0,1\}$-branching on binary variables. Here, we choose a subset $Q_{i,p}$ of variables such that the venue of team i is fixed in p by forcing $\sum_{x \in Q_{i,p}} x = 1$ or $\sum_{x \in Q_{i,p}} x = 0$. Hence, $Q_{i,p} := \{x_{i,j,p} \mid j \in T \setminus \{i\}\}$. Deciding the venue of a specific team in a specific period is part of the solution process for the problem introduced in section 2. Furthermore, solutions to LP relaxations to this problem do not imply a consistent decision about venues (which means there probably is a $i \in T$ and a $p \in P$ such that $\sum_{x \in Q_{i,p}} x \notin \{0,1\}$). Therefore, the branching idea is applicable as well as reasonable, here.

Defining a HAP set by branching can not guarantee binary solutions to the problem's LP relaxation. Table 2 represents a fractional solution on the left hand side. Each match in periods 1 and 2 has corresponding variable's value 0.5. Matches in period 3 have variable's value 1. The unique corresponding HAP set (having the minimum number of breaks) is shown on the right hand side. Hence, neither a $\sum_{x \in Q_{i,p}} x \notin \{0,1\}$, $i \in T$, $p \in P$, is given nor an integer solution is provided. However, we refuse to develop a full branching scheme in order to emphasize the power of the branching idea as a guiding scheme for the first levels of a branching tree. Consequently, if a node providing a full HAP set can not be pruned we solve the corresponding IP problem using the standard solver Cplex.

Period	1		2		3
MDs	1-2	1-4	2-1	2-3	2-4
	3-4	3-2	4-3	4-1	1-3
Variable	0.5	0.5	0.5	0.5	1

Team	1	2	3
1	0	1	0
2	1	0	0
3	0	1	1
4	1	0	1

Table 2: Fractional Solution Without Candidate for HAP set branching

Consider an IP problem corresponding to a HAP set where the path from the root node could not be pruned. In Briskorn [4] the corresponding LP relaxation is conjectured not to be able to provide a larger number of matches than the IP problem does. If this conjecture holds the LP relaxation corresponding to an infeasible HAP set is proofed infeasible and, consequently, the node is pruned before the IP problem is solved. Then, no IP problem corresponding to an infeasible HAP set has to be solved.

Let $\overline{Q_{i,p}} := \{x_{j,i,p} \mid j \in T \setminus \{i\}\}$. Then, $Q_{i,p} + \overline{Q_{i,p}} = 1$ due to (3) and, hence, $Q_{i,p} = 1 \Leftrightarrow \overline{Q_{i,p}} = 0$. Consequently, we implement each branching step by fixing subsets of variables $Q_{i,p}$ and $\overline{Q_{i,p}}$, respectively, to 0.

According to Miyashiro et al. [16] in a single RRT with the minimum number of breaks each team has no more than one break. If team i has no break at all we say it has a break in the first period which is justified by the fact that the first and the last entry of the corresponding HAP are identical. Therefore, we can specify each team's HAP by venue and period of its unique break. Consequently, we branch on venue and period of a specific team's break. Branching candidates are teams whose break is not fully specified by the optimal solution to the current node's LP relaxation.

We implement a specific break by setting match variables to 0 according to the break's venue and period. If we branch on team i to have a home-break in period p then we can set to zero half the match variables corresponding to i as follows.

$$\begin{aligned} Q_{i,p'} &= 0 \quad \forall p' \in P, ((p' < p \wedge p - p' \text{ even}) \vee (p' > p \wedge p' - p \text{ odd})) \\ \overline{Q_{i,p'}} &= 0 \quad \forall p' \in P, ((p' < p \wedge p - p' \text{ odd}) \vee (p' \geq p \wedge p' - p \text{ even})) \end{aligned}$$

Consequently, setting variables according to an away-break for team i in period p is done the other way round.

$$\begin{aligned} \overline{Q_{i,p'}} &= 0 \quad \forall p' \in P, ((p' < p \wedge p - p' \text{ even}) \vee (p' > p \wedge p' - p \text{ odd})) \\ Q_{i,p'} &= 0 \quad \forall p' \in P, ((p' < p \wedge p - p' \text{ odd}) \vee (p' \geq p \wedge p' - p \text{ even})) \end{aligned}$$

While we can represent team i having a break in p by fixing match variables (as seen above) we can not represent i not having a break in p by fixing match variables. Therefore, we propose a branching strategy where each subproblem is represented by a fixed break. Consequently, given a chosen branching candidate $i \in T$ we create a child node for each single break (defined by venue and period) which can be assigned to i. Taking into account that each team has exactly one break in a feasible solution we obtain a partition of solution space corresponding to the current node into solution spaces corresponding to its child nodes. Obviously, as lined out in Briskorn and Drexl [6] this means a number of up to $2(n-1)$ child nodes.

3.2 Achieving Feasible Home-Away-Pattern Sets

Avoiding infeasible HAP sets is substantial in order to save run times. As lined out in section 1 HAP sets can not easily proofed either feasible or infeasible. There is no simple characterization of feasible HAP sets. Below we propose several strategies to calculate the set of possible breaks (each defined by period and venue) for a given branching candidate $i \in T$. Since we construct a child node for each break the challenging part is to keep this set as small as possible without neglecting possible breaks.

No Restrictions: Here, we simply create a child node for each period and venue without consideration of breaks already fixed for other teams. Therefore, each node has $2(n-1)$ children if it is not pruned.

No Break Twice: The set of possible breaks can be reduced by taking into account the breaks already assigned to teams on the path from the root node to the current node. Two teams having identical breaks imply both teams having identical HAPs. A feasible HAP set can not contain two identical HAPs because the corresponding teams are not able to play against each other. Therefore, no break is assigned to more than one team. Consequently, the set of possible breaks can be reduced by all breaks being already assigned to a team on the path from the root node to the current node.

Break Sequences: It is known from, e.g., Miyashiro et al. [16] that there are either no or two breaks in each period. Hence, we have to choose $\frac{n}{2}-1$ periods (additional to the first period) where breaks occur and assign teams to both breaks corresponding to one of these periods in order to construct a HAP set. We refer to these periods as break periods in the remainder.

Given a HAP set a specific assignment of each HAP to a team does not influence the HAP set's feasibility. A HAP set with a minimum number of breaks is fully specified by a set of $\frac{n}{2}$ break periods as far as feasibility is concerned. The set of break periods in ascending order is referred to as break sequence in the remainder. In de Werra [9] break sequences corresponding to the special class of canonical 1-factorizations are studied. Here, we aim at identifying feasible general break sequences according.

In analogy to Briskorn and Drexl [6], let k and $\overrightarrow{V_k}$ denote the current node of the branching tree and the path from the root node to the current node. Additionally, let $nb_{p,\overrightarrow{V_k}}$ and $P^{br}_{\overrightarrow{V_k}}$ be the number of breaks fixed in period p on $\overrightarrow{V_k}$ and the set of periods already chosen as break periods on $\overrightarrow{V_k}$, respectively. Then, we can apply the following rules I to III as outlined in Briskorn and Drexl [6] in order to decide which breaks must be considered possible.

I A home-break (away-break) in the first period is possible if and only if no home-break (away-break) has been set in the first period on $\overrightarrow{V_k}$.

II If $nb_{p,\overrightarrow{V_k}}=1$ we can set a home-break (away-break) in period p if and only if the existing one is an away-break (home-break).

III We can set a break in period $p \in \{2,\ldots,n-1\}$, $nb_{p,\overrightarrow{V_k}}=0$ if and only if $\left|P^{br}_{\overrightarrow{V_k}} \setminus \{1\}\right| < \frac{n}{2}-1$.

Rule I states that a break in the first period is possible if this specific break has not been set on $\overrightarrow{V_k}$. Rule II takes care of the fact that in each period either two or no breaks are set. Hence, if exactly one break has been arranged in period p on $\overrightarrow{V_k}$ the complementary break is possible in p. Rule III decides whether a break can occur in a period where no break is arranged on $\overrightarrow{V_k}$ yet. Since there can be no more than $\frac{n}{2}$ periods having breaks (including the first period) a break in a period having no break so far is possible if less than $\frac{n}{2}-1$ periods (excluding the first period) have breaks already. Clearly, rules I and II cover "No Break Twice".

No Three Consecutive Breaks: As shown in Briskorn and Drexl [6] and Miyashiro et al. [17] we can further restrict the set of possible breaks: no three consecutive periods can be break

periods in a feasible HAP set with the minimum number of breaks. Therefore, we modify rule III of "Break Sequences" to rule III'.

III' We can set a break in period $p \in \{2, \ldots, n-1\}$, $nb_{p,\overrightarrow{V_k}} = 0$ if a break is possible in period p according to III and $\left(\left(nb_{p-2,\overrightarrow{V_k}} = 0 \vee nb_{p-1,\overrightarrow{V_k}} = 0\right) \wedge \left(nb_{p-1,\overrightarrow{V_k}} = 0 \vee nb_{p+1,\overrightarrow{V_k}} = 0\right) \wedge \left(nb_{p+1,\overrightarrow{V_k}} = 0 \vee nb_{p+2,\overrightarrow{V_k}} = 0\right)\right)$.

Rule III' checks whether a sequence of three consecutive break periods would be arranged if a break is fixed in period p. Note that we can further slightly strengthen III' as III".1 to III".5 by taking into account that there must be two breaks in the first period in the final HAP set no matter whether they are already set on $\overrightarrow{V_k}$ or not.

III".1 We can set a break in period $p \in \{4, \ldots, n-4\}$, $nb_{p,\overrightarrow{V_k}} = 0$ if a break is possible in period p according to III'.

III".2 We can set a break in period $p = 3$ if $nb_{3,\overrightarrow{V_k}} = 0$ and if a break is possible in period 3 according to III and $\left(\left(nb_{2,\overrightarrow{V_k}} = 0\right) \wedge \left(nb_{4,\overrightarrow{V_k}} = 0 \vee nb_{5,\overrightarrow{V_k}} = 0\right)\right)$.

III".3 We can set a break in period $p = 2$ if $nb_{2,\overrightarrow{V_k}} = 0$ and if a break is possible in period 2 according to III and $\left(nb_{n-1,\overrightarrow{V_k}} = 0 \wedge nb_{3,\overrightarrow{V_k}} = 0\right)$.

III".4 We can set a break in period $p = n-1$ if $nb_{n-1,\overrightarrow{V_k}} = 0$ and if a break is possible in period $n-1$ according to III and $\left(nb_{n-2,\overrightarrow{V_k}} = 0 \wedge nb_{2,\overrightarrow{V_k}} = 0\right)$.

III".5 We can set a break in period $p = n-2$ if $nb_{n-2,\overrightarrow{V_k}} = 0$ and if a break is possible in period $n-2$ according to III and $\left(\left(nb_{n-1,\overrightarrow{V_k}} = 0\right) \wedge \left(nb_{n-3,\overrightarrow{V_k}} = 0 \vee nb_{n-4,\overrightarrow{V_k}} = 0\right)\right)$.

Rule III".1 directly corresponds to III' for $p \in \{4, \ldots, n-4\}$. Special cases are periods 3, 2, $n-1$, and $n-2$ in rules III".2 to III".5 being strengthened in comparison to III'. Here, the first period is not checked for breaks because in a complete HAP set with the minimum number of breaks there are exactly two breaks in it.

Feasible Sequence: In the following we generalize the basic idea of "No Three Consecutive Breaks". First, we give the necessary condition for HAP sets to be feasible given in Miyashiro et al. [16]:

$$\sum_{p \in P} \min\{c_0(T', p), c_1(T', p)\} - \binom{|T'|}{2} \geq 0 \quad \forall T' \subseteq T \tag{9}$$

Condition (9) states that each subset T' of teams must be allowed to play $\binom{|T'|}{2}$ matches against each other. The number of matches between teams of T' in each period is restricted to the minimum of numbers of teams in T' playing at home and away, respectively. Choosing $l \in \left\{1, \ldots, \frac{n}{2}\right\}$ break periods means fixing $2l$ HAPs. In order to check (9) we have to take care of $2^{2l} - 2l - 1$ subsets of HAPs. Note that $|T'| = 2$ and $|T'| = 3$ are checked inherently

by "No Break Twice" and "No Three Consecutive Breaks", respectively. Here, we propose an efficient way to check all T' having exactly l HAPs and, therefore, $\binom{2l}{l}$ subsets of HAPs.

In order to establish a common notation we first introduce parts of the one in Miyashiro et al. [16]. We sort the given $2l$ HAPs as follows: We arrange two blocks of l HAPs each. The first (second) one consists of HAPs having 0 (1) in the last slot. Both blocks are ordered by ascending HAPs' break periods. An example with $2l = 6$ HAPs is shown on the left hand of table 3.

period	1	2	3	4	5
HAP 1	**0**	1	0	1	0
HAP 2	1	0	**0**	1	0
HAP 3	1	0	1	**1**	0
HAP 4	**1**	0	1	0	1
HAP 5	0	1	**1**	0	1
HAP 6	0	1	0	**0**	1

period	1	2	3	4	5
HAP 1	**1**	1	1	1	1
HAP 2	0	0	**1**	1	1
HAP 3	0	0	0	**1**	1
HAP 4	**0**	0	0	0	0
HAP 5	1	1	**0**	0	0
HAP 6	1	1	1	**0**	0

Table 3: Example for ordered HAP set and equivalent representation

The right hand side of table 3 provides an equivalent representation of the ordered (partial) HAP set as proposed in Miyashiro et al. [16]. The construction of this representation is done as follows.

- For HAPs having 0 in the last slot set all entries ahead of the break period to 0. Set the entry in the break period and all entries behind to 1.
- For HAPs having 1 in the last slot set all entries ahead of the break period to 1. Set the entry in the break period and all entries behind to 0.

Note that $\min\{c_0(T', p), c_1(T', p)\}$ is equivalent in corresponding columns p for each ordered set of HAPs T' and its equivalent representation, see Miyashiro et al. [16].

As special cases of T' Miyashiro et al. [16] introduce cyclically consecutive sets of HAPs and narrow sets of HAPs. A set of HAPs T' is narrow if and only if there is at least one period where all HAPs' entries are identical. Miyashiro et al. [16] show that given a HAP set for each subset of HAPs T' which is not narrow there is a narrow subset of HAPs T'' such that (9) is at least as tight for T'' as it is for T'. Consequently, checking (9) is restricted to narrow subsets of a HAP set.

Suppose a set $BP \subset P$, $|BP| = l - 1$, of break periods has been chosen on $\overrightarrow{V_k}$. In order to decide whether to consider an additional break period $p \in P \setminus \{1\}$, $nb_{p,\overrightarrow{V_k}} = 0$, to be possible or not we propose to check (9) for each subset of HAPs of cardinality l based on the set of break periods $BP \cup \{p\}$. Let $\mathcal{T}^l$ be the set of subsets of HAPs having exactly l HAPs and let $\mathcal{T}^{l'} \subset \mathcal{T}^l$ be the set of narrow subsets of HAPs having exactly one HAP for each break period.

Theorem 1. *For each $T' \in \mathcal{T}^l$ there is a $T'' \in \mathcal{T}^{l'}$ such that (9) is at least as tight for T'' as it is for T'.*

Proof. Given an arbitrary $T' \in \mathcal{T}^l \setminus \mathcal{T}^{l'}$ we construct $T'' \in \mathcal{T}^{l'}$ such that (9) is at least as tight for T'' as it is for T'. Circulate columns of T' such that there is no break in the first period. Consider the equivalent representation of T' as shown on the right hand side of table 3. Then, $c_0(T',1) = c_1(T',n-1)$ and $c_1(T',1) = c_0(T',n-1)$. If l is even there is at least one break period p such that $c_0(T',p-1) - 1 = c_0(T',p) = \frac{l}{2}$ or $c_0(T',p-1) + 1 = c_0(T',p) = \frac{l}{2}$. If l is odd there is at least one break period p such that $\min\{c_0(T',p-1), c_1(T',p-1)\} = \min\{c_0(T',p), c_1(T',p)\} = \lfloor \frac{l}{2} \rfloor$. Now, we construct T'' as follows:

- If l is even set $\frac{l}{2}$ ones and $\frac{l}{2}$ zeros in period p. Additionally, set $\frac{l}{2}+1$ ones and $\frac{l}{2}-1$ zeros in period $p-1$ by copying the pattern of p and exchanging one 0 by 1. If l is odd set $\lceil \frac{l}{2} \rceil$ zeros and $\lfloor \frac{l}{2} \rfloor$ ones in p. Additionally, set $\lceil \frac{l}{2} \rceil$ ones and $\lfloor \frac{l}{2} \rfloor$ zeros in $p-1$ by copying the pattern of p and exchanging one 0 by 1.
- Going backward from $p-1$ set a break in break period p' for an arbitrary HAP i having 0 in p' (then, i's entry in each period $p'' \in \{1,\ldots,p'-1\}$ is 1). Thus, the number of zeros is decreased by 1 at the predecessor of each break period. Proceed until the first period is reached or there is no 0 left in any HAP. If there is no 0 left in break period p' proceed by setting a break for an arbitrary HAP j not having a break yet in each break period (then, j's entry in each period $p'' \in \{1,\ldots,p'-1\}$ is 0). Thus, the number of zeros is increased by 1 at the predecessor of each break period $p'' \in \{1,\ldots,p'-1\}$.
- Going forward from p set a break in break period p' for an arbitrary HAP i having 1 in $p'-1$ (then, i's entry in each period $p'' \in \{1,\ldots,p'-1\}$ is 0). Thus, the number of zeros is increased by 1 in each break period. Proceed until the last period is reached or there is no 1 left in any HAP. If there is no 1 left in break period p' proceed by setting a break for an arbitrary HAP j not having a break yet in each break period (then, j's entry in each period $p'' \in \{1,\ldots,p'-1\}$ is 1). Thus, the number of zeros is decreased by 1 in each break period $p'' \in \{p',\ldots,n-1\}$.

First, we show that $T'' \in \mathcal{T}^{l'}$. Obviously, $T'' \in \mathcal{T}^l$. Let o_p be the ordinal of p within the break sequence (after circulating). If o_p is greater or equal $c_0(T',p)$ then there is at least one period $p' \in \{1,\ldots,p-1\}$ having no 0. If o_p is lower or equal $c_0(T',p)$ then $l - o_p$ is greater or equal than $c_1(T',p)$. Therefore, there is at least one period $p' \in \{p+1,\ldots,n-1\}$ having no 1. Thus, $T'' \in \mathcal{T}^{l'}$.

Second, we show that (9) is at least as tight for T'' as it is for T'.

Obviously, due to construction

$$\min\{c_0(T',p), c_1(T',p)\} = \min\{c_0(T'',p), c_1(T'',p)\}.$$

Note that, depending on T' the minimization term of (9) is increased by 1, is decreased by 1, or is not changed at all at each break period. The latter case appears at break period p' if two complementary HAPs with breaks in p' are contained in T'.

T'' does not contain any complementary HAPs by construction and, therefore, the minimization term of (9) is increased by 1 or is decreased by 1, respectively, at each break period. Hence,

$$\begin{aligned} \min\{c_0(T',p+k), c_1(T',p+k)\} &\geq \min\{c_0(T'',p+k), c_1(T'',p+k)\} \\ \forall k &\in \left\{-\frac{n-2}{2},\ldots,-1\right\} \cup \left\{1,\ldots,\frac{n-2}{2}\right\} \end{aligned}$$

(indices taken modulo $n-1$) since going backward and forward from p, respectively, the number of zeros or ones is strictly lowered to zero in T'' at each break period. $\square$

In order to illustrate the construction as done in the proof above we provide tables 4 to 6.

period	1	2	3	4	5	6	7
HAP 1	0	**0**	1	0	1	0	1
HAP 2	0	1	0	1	**1**	0	1
HAP 3	**0**	1	0	1	0	1	0
HAP 4	1	0	1	**1**	0	1	0
Min(0,1)	1	2	2	1	2	2	2

period	7	1	2	3	4	5	6
HAP 1	1	0	**0**	1	0	1	0
HAP 2	1	0	1	0	1	**1**	0
HAP 3	0	**0**	1	0	1	0	1
HAP 4	0	1	0	1	**1**	0	1
Min(0,1)	2	1	2	2	1	2	2

Table 4: T' before (left) and after (right) circulating

A HAP set (and, therefore, a subset of HAPs) T' is given on the left hand side of table 4. The HAPs' break periods differ from each other and, moreover, T' is not narrow. Therefore, $T' \in \mathcal{T}^l \setminus \mathcal{T}^{l'}$ with $l = 4$. First, we circulate periods until we have no break in the first period. This results into the set of HAPs shown on the right hand side of table 4 and does not affect the sum in term (9).

period	7	1	2	3	4	5	6
HAP 1	0	0	**1**	1	1	1	1
HAP 2	0	0	0	0	0	**1**	1
HAP 3	1	**0**	0	0	0	0	0
HAP 4	1	1	1	1	**0**	0	0
Min(0,1)	2	1	2	2	1	2	2

period	7	1	2	3	4	5	6
HAP 1	1	1	1	1	**0**	0	0
HAP 2	1	1	**0**	0	0	0	0
HAP 3	1	**0**	0	0	0	0	0
HAP 4	1	1	1	1	1	**0**	0
Min(0,1)	0	1	2	2	1	0	0

Table 5: Equivalent representation of T' (left) and T'' (right)

The left hand side of table 5 shows the equivalent representation of T'. We choose break period $p = 2$ since $c_0(T', 2-1)+1 = c_0(T', 2) = \frac{l}{2}$. Then, we construct T'' by reducing the number of zeros at each break period going backward from p to the first period and reducing the number of ones at each break period going forward from p to the last period. The result is shown on the right hand side of table 5. Note that $\min\{c_0(T', 2), c_1(T', 2)\} = \min\{c_0(T'', 2), c_1(T'', 2)\}$. Since the minimization term's value in (9) is strictly reduced at each break period going backward and forward from $p = 2$ in T'' $\min\{c_0(T', p), c_1(T', p)\} \geq \min\{c_0(T'', p), c_1(T'', p)\}$ for each $p \in P$.

On the left hand side of table 6 we illustrate the interpretation of the equivalent representation of T'' as a subset of HAPs. On the right hand side the set of HAPs is recirculated which finishes the construction of T''.

Thus, given a set of l break periods we can implicitly check condition (9) for each $T' \in \mathcal{T}^l$ by checking condition (9) for each $T'' \in \mathcal{T}^{l'}$ according to theorem 1. Note that $|\mathcal{T}^l| = \binom{2l}{l} = \frac{(2l)!}{l! \cdot l!} = \prod_{k=1}^{l} \frac{l+k}{k} > 2^l$ if $l > 1$ and $|\mathcal{T}^{l'}| = 2l$ which makes this reduction extremely useful.

Since each $T \in \mathcal{T}^{l'}$ is narrow there exists a period p_T such that from p_T to the following break period all HAPs of T have identical entries. Therefore, $T \in \mathcal{T}^{l'}$ can be specified by p_T

period	7	1	2	3	4	5	6
HAP 1	1	0	1	0	**0**	1	0
HAP 2	1	0	**0**	1	0	1	0
HAP 3	1	**1**	0	1	0	1	0
HAP 4	1	0	1	0	1	**1**	0
Min(0,1)	0	1	2	2	1	0	0

period	1	2	3	4	5	6	7
HAP 1	0	1	0	**0**	1	0	1
HAP 2	0	**0**	1	0	1	0	1
HAP 3	**1**	0	1	0	1	0	1
HAP 4	0	1	0	1	**1**	0	1
Min(0,1)	1	2	2	1	0	0	0

Table 6: T'' before (left) and after (right) recirculating

and the singleton entry in these periods. We can further reduce $\mathcal{T}^{l'}$ by eliminating subsets of HAPs being complementary to each other: condition (9) is equally tight for two subsets of HAPs $T_1 \in \mathcal{T}^{l'}$ and $T_2 \in \mathcal{T}^{l'}$ with $p_{T_1} = p_{T_2}$ and different entries in period p_{T_1} as shown in Miyashiro et al. [16]. Therefore, we check only the l subsets of HAPs having entry zero in periods where all entries are identical. As shown in Miyashiro et al. [17] each check can be done in linear time.

The test above is applied in addition to "No Three Consecutive Breaks". We apply it only for $l \geq 4$ since "No Three Consecutive Breaks" covers $l < 4$.

Feasible Subsequences: Note that $\mathcal{T}^l \subset 2^T$. Hence, checking condition (9) for each $T' \in \mathcal{T}^l$ as proposed by "Feasible Sequence" can not ensure condition (9) for each $T' \subseteq T$. For an example consider the partial HAP set h illustrated in table 7.

period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
HAP 1	0	1	0	1	0	**0**	1	0	1	0	1	0	1	0	1
HAP 2	0	1	0	1	0	1	0	1	**1**	0	1	0	1	0	1
HAP 3	0	1	0	1	0	1	0	1	0	**0**	1	0	1	0	1
HAP 4	0	1	0	1	0	1	0	1	0	1	**1**	0	1	0	1

Table 7: Infeasible subsequence

Checking break sequence $\{6, 9, 10, 11\}$ according to "Feasible Sequence" does not identify h to be infeasible. Clearly, break sequence $\{9, 10, 11\}$ would be identified as infeasible but is never checked according to "Feasible Sequence" if $p = 11$ is the break period being added last (in the course of our branching scheme). In order to fix this flaw we propose to check subsequences of break sequences, as well.

According to Miyashiro et al. [16] not each subset of HAPs must be checked if the minimum number of breaks is required. We have to check only those subsets having no more than $\frac{n}{2}$ HAPs and exclusively containing cyclically consecutive HAPs. We aim at implicitly checking all these subsets of HAPs by checking subsequences. We need to check only cyclically consecutive subsequences of break periods since subsequences not being cyclically consecutive correspond to sets of HAPs not being cyclically consecutive. Note that there exist exactly two narrow and cyclically consecutive sets of HAPs corresponding to a given cyclically consecutive subsequence of break periods. These two sets of HAPs are complementary to each other and, hence, we have to check only one set of HAPs for each subsequence.

According to our branching scheme we add break periods step by step. Therefore, we can restrict checking subsequences to subsequences incorporating the new break period in each

step. Subsequences not containing the new break period have been checked before. Again, checking a specific (sub-)sequence of break periods is done according to Miyashiro et al. [17] in linear time.

3.3 Choice of Branching Candidates

The branching scheme prescribes to create child nodes corresponding to each possible break of a specific team. Consequently, branching candidates correspond to teams. We propose several strategies to choose the branching candidate below. First, we define a fractional break value $br'_{i,p} = |1 - \sum_{j \in T \setminus \{i\}} (x_{i,j,p-1} + x_{i,j,p})|$ where $x_{i,j,0}$ means $x_{i,j,n-1}$ for each team $i \in T$ and period $p \in P$ according to the current node's optimal solution.

Random Choice: Choosing the branching team randomly minimizes computational effort. Here, we implement a (pseudo-)random selection by choosing the team having index $d_k + 1$ where d_k is the depth of the current node k.

Largest Fractional Break: We choose team $i' = \arg\max_i \left\{\max_p \left\{br'_{i,p} \mid br'_{i,p} < 1\right\}\right\}$ with largest fractional break value as branching team. The motivation for this strategy is as follows. First, the current node's optimal solution is cut by fixing i''s break. Second, i' having this specific break is likely to enable low cost tournaments due to cost orientation of the LP relaxation.

Most Infeasible 1: Among all teams we choose the one having the most infeasible constellation of break values. The common idea is to enforce feasibility for those teams first where least tendency is given by the solution to the LP relaxation which break to choose, see Achterberg et al. [1]. Note that not each team must have at least one break (including those in the first period) in a feasible solution to the LP relaxation. Accordingly, a feasible solution to the LP relaxation might provide a team i with $\sum_{p \in P} br'_{i,p} > 1$. Since this effect severely contradicts the structure of a feasible IP solution we choose team $i' = \arg\max_i \left\{\sum_{p \in P} br'_{i,p}\right\}$ as branching team if $\sum_{p \in P} br'_{i',p} > 1$. Otherwise, we choose the branching team according to "Largest Fractional Break".

Most Infeasible 2: We choose team $i' = \arg\min_i \left\{\max_p \{br'_{i,p}\}\right\}$. The basic idea is the same as for "Most Infeasible 1" but infeasibility is measured differently. There can be two reasons for the maximum fractional break value being small. First, $\sum_{p \in P} br'_{i',p}$ is small (in particular $\sum_{p \in P} br'_{i',p} < 1$). Second, $\sum_{p \in P} br'_{i',p} = 1$ but fractional break values $br'_{i',p}$ are larger than 0 for many periods p. Both effects contradict the structure of a feasible IP solution.

3.4 Node Order Strategy

We consider "Depth First Search" and "Breadth First Search" as node order strategies. We observe that "Breadth First Search" is significantly more efficient due to less nodes being explored for instances with less than 10 teams. For larger instances our list of nodes grows that large that administrative effort uses up this advantage. Therefore, we propose a compromise between a short node list guaranteed by "Depth First Search" and a small number of nodes being explored guaranteed by "Breadth First Search" based on the father node's optimal LP solution.

First, we obtain v_{cur} as normalized current node's solution value dividing it by the lower bound. Second, we calculate the fraction x_{bin} of match variables having binary values in the current node's optimal solution. Then, we sort the node list in ascending order of $v_{cur} - w \cdot x_{bin}$ where $w \in \mathbb{R}$. The idea is to explore a node earlier if its father's optimal solution provides many binary variables. In preliminary tests $w = 1.5$ proofed to be a good choice as far as running times are concerned.

According to the strategies proposed above all children of a single node are considered identical. Therefore, we additionally have to decide in which order nodes having the same father are explored. We propose several strategies hereafter.

Pseudo Cost: We calculate pseudo cost $c^h_{i,p}$ and $c^a_{i,p}$ representing average cost of matches being possible if i is branched to have a home-break or away-break, respectively, in period p.

$$\begin{aligned} c^h_{i,p} &= \frac{1}{(n-1)^2} \sum_{j \in T \setminus \{i\}} \left(\sum_{p'=1}^{\lfloor \frac{p-1}{2} \rfloor} c_{j,i,p-2p'} + \sum_{p'=1}^{\lceil \frac{p-1}{2} \rceil} c_{i,j,p+1-2p'} \right) + \\ & \frac{1}{(n-1)^2} \sum_{j \in T \setminus \{i\}} \left(\sum_{p'=1}^{\lceil \frac{n-p}{2} \rceil} c_{i,j,p-2+2p'} + \sum_{p'=1}^{\lfloor \frac{n-p}{2} \rfloor} c_{j,i,p-1+2p'} \right) \end{aligned} \tag{10}$$

$$\begin{aligned} c^a_{i,p} &= \frac{1}{(n-1)^2} \sum_{j \in T \setminus \{i\}} \left(\sum_{p'=1}^{\lfloor \frac{p-1}{2} \rfloor} c_{i,j,p-2p'} + \sum_{p'=1}^{\lceil \frac{p-1}{2} \rceil} c_{j,i,p+1-2p'} \right) + \\ & \frac{1}{(n-1)^2} \sum_{j \in T \setminus \{i\}} \left(\sum_{p'=1}^{\lceil \frac{n-p}{2} \rceil} c_{j,i,p-2+2p'} + \sum_{p'=1}^{\lfloor \frac{n-p}{2} \rfloor} c_{i,j,p-1+2p'} \right) \end{aligned} \tag{11}$$

Child nodes are explored in ascending order of pseudo cost $c^h_{i,p}$ and $c^a_{i,p}$ corresponding to breaks at home and away, respectively, of branching team i in period p. Since $c^h_{i,p}$ and $c^a_{i,p}$ are static we calculate them once before the B&B procedure starts and, therefore, computational effort is small.

Pseudo Cost Revisited: If we branch on team i to have a specific break pseudo costs according to other teams may be altered in the resulting branching subtree. Suppose, for example, team i is branched to have a home-break in period 2 and j is chosen as branching candidate in the corresponding subtree. In pseudo-cost $c^a_{j,4}$ cost $c_{j,i,2}$ and $c_{i,j,3}$ should not be considered (in contrast to calculation in (11)) since these matches are not possible due to the fixed break of i.

More generally, suppose team i is branched to have a break in period p. Additionally, a break for branching candidate j in period p' is considered in the subtree. We eliminate

- cost according to a match between teams i and j in period $p'' \in \{\min\{p,p'\}, \ldots, \max\{p,p'\} - 1\}$ from pseudo cost according to j and period p' if either i's and j's break venues are identical and $|p - p'|$ is odd or the break venues differ and $|p - p'|$ is even,

- cost according to a match between teams i and j in period $p'' \in \{1, \ldots, \min\{p, p'\} - 1\} \cup \{\max\{p, p'\}, \ldots, n-1\}$ from pseudo cost according to j and period p' if either i's and j's break venues are identical and $|p - p'|$ is even or the break venues differ and $|p - p'|$ is odd.

Then, child nodes are explored in ascending order of revisited pseudo cost.

Largest Fractional Break First: We first explore the child node corresponding to the branching team i's break (defined by period and venue) which implies the largest fractional break value $br'_{i,p}$ according to the father node's optimal solution. The remaining child nodes are explored in ascending order of "Pseudo Cost" or "Pseudo Cost Revisited", respectively. The idea is here that a large fractional break value might indicate a profitable break due to the LP problem's objective of cost minimization.

4 Computational Results

First, we evaluate the strategies in order to avoid infeasible HAP sets. We choose branching candidates according to "Random Choice". Furthermore, we employ depth first search and insert child nodes in ascending order of the corresponding breaks' periods. Since the resulting branching scheme is static and has no cost orientation, differences in the number of nodes being explored as well as run times are exclusively caused by decisions whether a node corresponding to a specific break is constructed or not.

Strategies "No Restrictions", "No Break Twice", "Break Sequences", and "No Three Consecutive Breaks" are carried out for 8 teams since the run times grow too high for instances with more teams as long as no more efficient strategy is employed. Results are given in relation to results obtained using "No Restrictions". Evaluation of strategy "Feasible Sequence" as described in section 3.2 and a special case, namely "Four Breaks in Five Periods", is done solving instances with 10 teams. Here, results are given in relation to results obtained using "No Three Consecutive Breaks" for $n = 10$. Each strategy is employed in addition to the previous ones except "Four Breaks in Five Periods" replacing "Feasible Sequence". We focus on the decreasing number of LP problems to be solved (LP red.) and infeasible LP problems (i.LP red.), respectively. Furthermore, the reduction of average run times (r.t. red.) is given in table 8.

As we can see the number of nodes being explored is strictly decreased by each strategy. Using "No Break Twice" and "Break Sequences" reduces the number of LP problems by 50%. More remarkably, the number of infeasible LP problems is reduced by more than 85%. Note that both strategies avoid constructing infeasible (partial) HAPs which would immediately result into an infeasible LP problem. "No Three Consecutive Breaks" takes a step further by avoiding infeasible (partial) HAPs not corresponding to an infeasible LP problem but corresponding to an infeasible subtree. Here, we omit solving these LP problems and, possibly, further child nodes. Applying "No Three Consecutive Breaks" we do not obtain a single infeasible LP problem. The overall number of LP problems and run times are reduced by more than 70% and more than 50%, respectively.

Considering 10 teams nearly each infeasible LP problem is avoided if "Feasible Sequence" is applied. Here, we recognize a special case for 10 teams. Four break periods in a sequence of five periods (without three consecutive break periods) is not infeasible in general but for

Strategy	LP red.	i.LP red.	r.t. red.
No Restrictions ($n = 8$)	—	—	—
No Break Twice ($n = 8$)	26.1%	48.7%	6.7%
Break Sequences ($n = 8$)	49.4%	87.3%	22.9%
No Three Consecutive Breaks ($n = 8$)	72.7%	100.0%	54.6%
No Three Consecutive Breaks ($n = 10$)	—	—	—
Feasible Sequence ($n = 10$)	30.0%	99.5%	28.9%
Four Breaks in Five Periods ($n = 10$)	33.0%	100.0%	32.1%

Table 8: Comp. Results for Branching Strategies

$n = 10$. The fifth break period can not be feasibly chosen according to "Feasible Sequence". In order to fine-tune our approach we consider strategy "Four Breaks in Five Periods" and observe that no infeasible LP problem remains to be solved. Both the overall number of LP problems and run times are reduced by more than 30%. Note that "Feasible Subsequences" is not evaluated, here, since this strategy takes effect only for problems with more than 10 teams. However, tests for larger problems are not possible due to run times. For the sake of completeness we line out that the average number of IP problems which are solved for each problem instance is 0.9 for $n = 8$ and 4.7 for $n = 10$.

Among strategies to choose the branching candidate "Most Infeasible 1" turned out to be the most efficient. Additionally, we employ "Breadth First Search" considering the number of binary variables and "Pseudo Cost Revisited" as node order strategies. Out of the strategies evaluated above "No Three Consecutive Breaks" combined with "Four Breaks in Five Periods" is applied here. Run times are given in table 9.

n	B&B	Cplex
6	0.15	0.23
8	12.62	26.79
10	7841.14	—

Table 9: Comp. Results for Minimum Number of Breaks

Clearly, our B&B approach outperforms Cplex using default settings for $n < 10$. Cplex runs out of memory after about 12 hours for $n = 10$. Note that Cplex was employed using "Depth First Search" to overcome lack of memory in Briskorn and Drexl [5, 6]. Not even a feasible solution is found within 6 days of running time, then. Accordingly, proofing optimality in 2.2 hours on average clearly indicates superiority of our approach. The optimal solution was found in less than 24 minutes on average.

5 Conclusion

We propose a branch-and-bound approach in order to find minimum cost RRT schedules. The basic idea originates from a decomposition scheme fixing each team's venue in each period, first, and arranging matches, afterwards. An outstanding step towards run times savings is to anticipate infeasible HAP sets on the basis of partial HAP sets as done by "No Three Consecutive Breaks" and following strategies.

Since fixing each team's venue in each period by branching on breaks can not guarantee integer solutions we have to handle the case where no branching candidate is given for a fractional solution. We propose to solve the corresponding IP problem using Cplex. Since solving IP problems is significantly more time consuming than solving LP problems we emphasize that for all test instances only a very small number of IP problems had to be solved. Our branching scheme is likely to obtain HAP sets that are feasible. Consequently, most probably only IP problems corresponding to feasible HAP sets have to be solved. In fact, no infeasible IP problem was solved in our test runs.

For further research we can think of incorporating several aspects having relevance in real world tournaments and being related to HAP sets. For example, stadium availability can not be taken for granted since most stadiums are home venue of more than one team. Additionally, teams have preferences when to play at home and when to play away. There are leagues where no break must occur in specific periods. It is easy to see, that such requirements can be considered in the set of possible breaks.

Beside integer programming constraint programming and hybrid approaches have been successfully applied in order to solve sports leagues problems, see Rasmussen and Trick [18] for example. In many approaches choosing a HAP set is critical since feasibility can not be checked immediately. Here, our tests of our strategies might provide tools in order to efficiently ensure several necessary conditions.

References

[1] T. Achterberg, T. Koch, and A. Martin. Branching Rules Revisited. *European Journal of Operational Research*, 33:42–54, 2005.

[2] T. Bartsch. *Sportligaplanung – Ein Decision Support System zur Spielplanerstellung (in German).* Deutscher Universitätsverlag, Wiesbaden, 2001.

[3] T. Bartsch, A. Drexl, and S. Kröger. Scheduling the Professional Soccer Leagues of Austria and Germany. *Computers & Operations Research*, 33:1907–1937, 2006.

[4] D. Briskorn. Feasibility of Home–Away–Pattern Sets: A Necessary Condition. *Working Paper*, 2007.

[5] D. Briskorn and A. Drexl. Scheduling Sports Leagues using Branch-And-Price. *Working Paper*, 2006.

[6] D. Briskorn and A. Drexl. Branching Based on Home–Away–Pattern Sets. In *GOR Proceedings 2006*. Springer, Berlin, Germany. Forthcoming.

[7] D. Briskorn, A. Drexl, and F. C. R. Spieksma. Round Robin Tournaments and Three Index Assignment. *Working Paper*, 2006.

[8] P. Brucker and S. Knust. *Complex Scheduling*. Springer, Berlin, 2006.

[9] D. de Werra. Geography, Games and Graphs. *Discrete Applied Mathematics*, 2:327–337, 1980.

[10] D. de Werra. Scheduling in Sports. In P. Hansen, editor, *Studies on Graphs and Discrete Programming*, pages 381–395. North-Holland, Amsterdam, The Netherlands, 1981.

[11] D. de Werra. Minimizing Irregularities in Sports Schedules Using Graph Theory. *Discrete Applied Mathematics*, 4:217–226, 1982.

[12] D. de Werra. On the Multiplication of Divisions: The Use of Graphs for Sports Scheduling. *Networks*, 15:125–136, 1985.

[13] D. de Werra. Some Models of Graphs for Scheduling Sports Competitions. *Discrete Applied Mathematics*, 21:47–65, 1985.

[14] D. de Werra, T. Ekim, and C. Raess. Construction of Sports Schedules with Multiple Venues. *Discrete Applied Mathematics*, 154:47–58, 2006.

[15] A. Drexl and S. Knust. Sports League Scheduling: Graph- and Resource–Based Models. *Omega*, 35:465–471, 2007.

[16] R. Miyashiro, H. Iwasaki, and T. Matsui. Characterizing Feasible Pattern Sets with a Minimum Number of Breaks. In E. Burke and P. de Causmaecker, editors, *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 2740, pages 78–99. Springer, Berlin, Germany, 2003.

[17] R. Miyashiro, H. Iwasaki, and T. Matsui. Characterizing Feasible Pattern Sets with a Minimum Number of Breaks. In E. Burke and P. de Causmaecker, editors, *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, pages 311–313. Springer, Berlin, Germany, 2003.

[18] R. V. Rasmussen and M. A. Trick. A Benders Approach for the Constrained Minimum Break Problem. *European Journal of Operational Research*, 177:198–213, 2007.

[19] J. A. M. Schreuder. Constructing Timetables for Sport Competitions. *Mathematical Programming Study*, 13:58–67, 1980.

[20] J. A. M. Schreuder. Combinatorial Aspects of Construction of Competition Dutch Professional Football Leagues. *Discrete Applied Mathematics*, 35:301–312, 1992.