Briskorn, Dirk

Working Paper

A branching scheme for minimum cost tournaments with regard to real world constraints

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 643

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

Suggested Citation: Briskorn, Dirk (2009) : A branching scheme for minimum cost tournaments with regard to real world constraints, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 643, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/147561

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 643

A Branching Scheme for Minimum Cost Tournaments with regard to Real World Constraints

Dirk Briskorn

February 2009

Dirk Briskorn
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

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. Moreover, stadium availability, fixed matches, and regions' capacities are taken into account. The goal is to find the minimum cost tournament among those having the minimum number of breaks and being feasible with regard to these additional constraints. A branching scheme is developed where branching is done by fixing a break for a team. Thus, the focus is on identifying breaks leading to an infeasible home away pattern set in order to avoid the resulting infeasible subtrees of a branch and bound tree.

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

1 Introduction

Round robin tournaments (RRT) cover a huge variety of different types of sports league schedules arising in practice. In a single RRT a set of teams T, $|T| = n$ even, competes such that each team plays exactly once against each other team, either at home or at the opponent's home. In the latter case, we say the first team plays away. Furthermore, each team has to play exactly once in each period and, hence, we have a set P of $n-1$ periods altogether.

A mirrored double RRT is a tournament where each pair of teams meet twice, both, the first half and the second half of the tournament, form a single RRT each, and team i plays at home against team j in period p of the first half if and only if j plays at home against team i in period p of the second half. It is easy to see that we can schedule a mirrored double RRT by projecting all requirements regarding the second half on the first half and, then, schedule a single RRT.

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 $p > 1$ if and only if i plays either twice at home or twice away 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 cannot be less than $n-2$ in a single RRT. Furthermore, de Werra [9] shows that this number of breaks can be obtained for each even n. The single RRT provided in table 1 has 4 breaks for 6 teams and, therefore, it has the minimum number of breaks.

The constraints introduced above are due to the type of tournament. In the following, we will outline constraints arising in the context of real world tournaments. First, an important restriction is caused by stadium unavailability. A team i's stadium may not be available in some periods and, consequently, i has to play at the opponent's venue then. Suppose for example, that teams 3 and 4 cannot play at home in period 4 and 2, respectively. Then, the single RRT in Table 1 is not feasible but the one in Table 2 is (and, additionally, it has a minimum number of breaks also).

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

Table 2: Single RRT for $n = 6$

Obviously, stadium availability just serves as an example for causes forcing teams to play in the opponent's stadium in a specific period. For example, it can be the organizers goal to fulfill teams' preferences to play at home or away. Preferences are mostly due to individual interests, e.g. german soccer club Bayern Munich wants to play at home during the Oktoberfest. In the context of a mirrored double RRT, a team forced to play away (due to stadium unavailability) in period p of the second half is forced to play at home in period p of the first half. That is why we consider this kind of restriction, namely a team has to play at home in a certain period, as well.

A second constraint is given by fixed matches which is a common requirement in real world tournament scheduling, see Bartsch et al. [2] for example. If a match of i against j is fixed to be carried out in period p, then we require that both teams compete in p no matter at which venue. Of course, combining this requirement with stadium unavailability for team i in p, for example, we can fix the venue to be j's home, as well. Projection of requirements in the second half to the first half is straightforward.

A third restriction concerns groups of teams from the same area. A prominent example is London where no less than 13 professional teams are located. Five of them played in the Premier League, England's first soccer league, in season 2005/2006. Even if each of these teams has a stadium on its own the infrastructure of the region might be overloaded if too many teams play at home in a specific period. For example, traffic jams and overcrowded public transportation systems resulting from fans heading to the stadiums at the same time must be avoided. Furthermore, the capacity of security staff and of firemen needed in case of emergency is limited. Consequently, the number of matches teams of this area play at home in the same period should be limited. Since we consider a mirrored double RRT we have to limit the number of matches teams of this area play away in the same period as well. This is because these matches correspond to matches at home in the second half. Suppose for example that teams 1, 3, and 4 are located in the same region and are not allowed to play at home in the same period. In the single RRT represented by Table 1 this requirement is violated in period 4. Additionally, in period 3 all of these teams play away which results in 3 matches at home in period 3 of the second half. The single RRT in Table 2 fulfills this requirement.

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 [16]. Brucker and Knust [8] and Drexl and Knust [16] 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. Real world problems have been considered in Bartsch et al. [2], Della Croce and Oliveri [15], Durán et al. [17], Kendall [19], Rasmussen [22], Ribeiro and Urrutia [24] and Schreuder [25] examine particular formulations.

Extensive overviews of literature on sports leagues scheduling in the context of operations research are provided by Knust [20] and Rasmussen and Trick [23].

In Briskorn and Drexl [5] a prominent decomposition scheme is picked up as the basic idea for a branch&bound (B&B) approach for scheduling single RRTs with a minimum number of breaks (but without taking stadium availability and regions' capacities into consideration). Using the linear programm (LP) relaxation of an integer programming model branching is done by fixing a break for a team. Doing this the venues of all teams are fixed step by step. It is well known, see Miyashiro et al. [21] for example, that we cannot fix venues arbitrarily since there may be no corresponding single RRT. Thus, the focus is on identifying breaks leading to infeasible combinations of venues and, then, ignoring the corresponding subtrees. Briskorn and Drexl [5] develop several efficient tests to identify infeasible breaks. The scheme has been proven to be capable to significantly reduce the number of nodes related to an infeasible linear programm (LP) and it clearly outperforms state-of-the-art solver Cplex. Therefore, in the paper at hand we aim at extending the approach such that stadium unavailibility, fixed matches, and regions' constraints can be taken into account.

In the remainder 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 the branching scheme. Section 4 lines out computational results and, finally, Section 5 contains conclusions and an outlook to further research.

2 Problem Definition and Model

Given a set of teams T, $|T| = n$ even and a set of periods P, $|P| = n - 1$, a schedule S can formally specified as a subset of $T \times T \times P$. Each triple $(i, j, p) \in T \times T \times P$ with $i \neq j$ corresponds to a match of team i against team j at i's home in period p. We say i plays at home against j in p according to S and j plays away against i in p according to S if $(i, j, p) \in S$.

A subset $S \subset T \times T \times P$ is a schedule if each team plays exactly once against each other team and each team plays exactly once per period according to S. We say team i has a home break and an away break in period $p > 1$ if and only if i plays twice at home and twice away, respectively, in periods $p - 1$ and p. A string specifying the venue (home/away) of a team for each period is called a home away pattern (HAP).

We use the following notation in order to specify stadium availability. We have sets $H_p \subset T$ and $A_p \subset T$ for each $p \in P$. If $i \in H_p$ and $i \in A_p$, then i must play at home and away, respectively, in period p. Note that teams in $T \setminus (H_p \cup A_p)$ can play both, at home and away, in p. Consequently, a schedule S is feasible regarding stadium availability constraints if

- for each period $p \in P$ each team $i \in H_p$ and $i \in A_p$ plays at home and away, respectively, in p according to S.

Fixed matches are particularized using the notation described in the following. For each period $p \in P$ we have a set $F_p \subset \{(i,j) \mid i,j \in T, i < j\}$ of all pairs of teams such that $(i,j) \in F_p$ if and only if i must compete j in p. For notational convenience $F = \bigcup_{p \in P} F_p$. Note that $F_p \cap F_{p'} = \emptyset$ if $p \neq p'$ and for each team $i \in T$ and each period $p \in P$ there cannot be more than one pair of teams containing i in F_p. Consequently, a schedule S is feasible regarding fixed matches if

II for each period $p \in P$ and each $(i,j) \in F_p$ i plays against j in p according to S.

We use the following notation in order to specify constraints regarding regions. We have a set $\mathcal{R} \subseteq 2^T$ of regions. Each region $R \in \mathcal{R}$ has capacity C_R which specifies the maximum number of matches allowed to be carried out at home simultaneously and, thus, the maximum number of matches allowed to be carried out away simultaneously by teams in R. A schedule is feasible regarding regions' capacities if

III for each period $p \in P$ and each region $R \in \mathcal{R}$ the number of matches of teams in R carried out at home does not undershoot $|R| - C_R$ and does not exceed C_R according to S.

A feasible schedule is a schedule providing a minimum overall number of breaks and fulfilling I to III. Furthermore, we associate cost $c_{i,j,p}$ with (i,j,p). 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 and Drexl [6]. We define the cost of schedule S as the sum of costs corresponding to triples in S. Then, we consider the problem to find a feasible schedule S with minimum cost, namely the minimum cost single RRT problem with side constraints (MCSRRT).

MCSRRT can be represented by an IP model as specified by (1) to (12). 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) to (4) assure the single RRT structure. Restrictions (2) and (3) force each pair of teams to meet exactly once. This can happen in an arbitrary period if (i,j) is not fixed and in the predefined period if $(i,j) \in F_p$. Constraint (4) forces each team to play exactly once per period. Then, restrictions (5) or (6) ensure that $br_{i,p}$ equals 1 if team i plays twice at home or twice away, respectively, in periods $p-1$ and p. Constraint (7) limits the number of breaks to the minimum number. Note that combining (5), (6), and (7) leads to $br_{i,p} = 1$, $i \in T$, $p \in P^{\geq 2}$, if and only if i has a break in p. Constraint (8) enforces that matches are arranged regarding stadium availabilities. Constraints (9) and (10) limit the number of matches in region R to no more than the corresponding capacity C_R.

In this paper we additionally consider three special cases of problem MCSRRT. These are derived by setting $H_p = A_p = \emptyset$ for each $p \in P$, $F_p = \emptyset$ for each $p \in P$, or $\mathcal{R} = \emptyset$. More specifically, we tackle the minimum cost single RRT problem with stadium unavailabilities (MCSRRTs) where $F_p = \emptyset$ for each $p \in P$ and $\mathcal{R} = \emptyset$, the minimum cost single RRT problem with fixed matches (MCSRRTf) where $H_p = A_p = \emptyset$ for each $p \in P$ and $\mathcal{R} = \emptyset$, and the minimum cost single RRT problem with regions (MCSRRTr) where $H_p = A_p = F_p = \emptyset$ for each $p \in P$ and $\mathcal{R} = \emptyset$. Thus, in each of these special cases only one of I, II and III is to be considered.

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

$$\begin{aligned}
\text{s.t.} \sum_{p \in P} (x_{i,j,p} + x_{j,i,p}) &= 1 && \forall\, i,j \in T, i < j, (i,j) \notin F && (2)\\
x_{i,j,p} + x_{j,i,p} &= 1 && \forall\, p \in P, (i,j) \in F_p && (3)\\
\sum_{j \in T \setminus \{i\}} (x_{i,j,p} + x_{j,i,p}) &= 1 && \forall\, i \in T, p \in P && (4)\\
\sum_{j \in T \setminus \{i\}} (x_{i,j,p-1} + x_{i,j,p}) - br_{i,p} &\leq 1 && \forall\, i \in T, p \in P^{\geq 2} && (5)\\
\sum_{j \in T \setminus \{i\}} (x_{j,i,p-1} + x_{j,i,p}) - br_{i,p} &\leq 1 && \forall\, i \in T, p \in P^{\geq 2} && (6)\\
\sum_{i \in T} \sum_{p \in P^{\geq 2}} br_{i,p} &\leq n-2 && && (7)\\
x_{i,j,p} &= 0 && \forall\, p \in P, i \in A_p \vee j \in H_p && (8)\\
\sum_{i \in R} \sum_{j \in T \setminus \{i\}} x_{i,j,p} &\leq C_R && \forall\, p \in P, R \in \mathcal{R} && (9)\\
\sum_{i \in R} \sum_{j \in T \setminus \{i\}} x_{j,i,p} &\leq C_R && \forall\, p \in P, R \in \mathcal{R} && (10)\\
x_{i,j,p} &\in \{0,1\} && \forall\, i,j \in T, i \neq j, p \in P && (11)\\
br_{i,p} &\in \{0,1\} && \forall\, i \in T, p \in P^{\geq 2} && (12)
\end{aligned}$$

3 Branch–and–Bound Algorithm

3.1 Basic Idea

As in Briskorn and Drexl [5] the basic idea is to use the LP relaxation of the model presented in Section 2 and to branch on the break of teams. Note that there is exactly one break per team if we consider teams without a break to have a break in the first period, see Miyashiro et al. [21]. At each node of the search tree we first solve the LP relaxation of (1) to (12). Regarding the found solution, a team i is chosen whose break is not determined yet. Then, for each possible break for i a child node is created where the corresponding break is fixed for i. By fixing the break for i we decide for each period whether i plays at home or away. Note that the HAP is strictly alternating in all periods but the unique break period. Unfortunately, there may be up to $2(n-1)$ child nodes since a break can occur in each period and either at home or away. Hence, the search tree can grow extremely broad. Here, the challenging part is to reduce the set of possible breaks for i and, therefore, the number of child nodes of the current node dynamically as much as possible without cutting out feasible solutions.

A path in the search tree from the root node to a node with depth n defines a HAP set that is a set of HAPs assigned to teams. A HAP set is called feasible if there is a single RRT where each team plays in each period according to the HAP set. The complexity of deciding feasibility of a HAP set is an open question in the general case as well as in the case where the

number of breaks is $n-2$, see Briskorn [3] and Miyashiro et al. [21]. When deciding whether to create or not a child node (corresponding to a specific break) we have to decide whether the resulting partial HAP set can be completed to a feasible full HAP set. In the following we call a partial HAP set feasible if and only if there is a corresponding feasible full HAP set. Obviously, if we know that a partial HAP set is infeasible, then we can prune the corresponding subtree. Furthermore, for a specific node we call a break infeasible if it leads to an infeasible partial HAP set. Then, we ignore infeasible breaks when creating child nodes.

Briskorn and Drexl [5] consider the model consisting of (1) to (7) as well as (11) and (12) where $F = \emptyset$ and develop necessary conditions for a specific break to be feasible. In order to keep the paper at hand self-contained we first state these conditions. We assume a set P^{br} of periods has already been chosen as break periods for a set T^{br} of teams on the path from the root node to the current node of the search tree. Then, a specific break defined by period p and venue for team $i \in T \setminus T^{br}$ is infeasible if

1. the resulting set $P^{br} \cup \{p\}$ of break periods contains $n/2+1$ periods or
2. a subset $\hat{T}^{br} \subseteq T^{br} \cup \{i\}$ of teams cannot play $\binom{|\hat{T}^{br}|}{2}$ matches among each other.

These conditions are motivated by two well known properties of feasible HAP sets, namely

1. in each period there are either two or zero breaks and, therefore, there are exactly $n/2$ break periods and
2. each subset $\hat{T} \subseteq T$ of teams must play $\binom{|\hat{T}|}{2}$ matches against each other since each team must play against each other.

Details regarding these conditions can be found in Miyashiro et al. [21]. In Briskorn and Drexl [5] efficient tests for the conditions above are implemented. Note that feasibility of a specific break depends neither on team i nor on T^{br} if stadium unavailability, fixed matches, and regions' capacities are not considered. Feasibility of a specific break can be decided based on P^{br} and p only since – regarding feasibility of the partial HAP set – teams are interchangable.

In the paper at hand, however, we aim at strengthening the tests mentioned above for the case where at least one of stadium unavailability, fixed matches, and regions' capacities has to be considered using individual information about teams. Therefore, we extend the concept of infeasibility of a specific break with regard to a given set of break periods to infeasibility of a specific break for a specific team with regard to a given set of breaks assigned to a set of teams. That is, depending on the set of teams involved a partial HAP set may be feasible or not.

Consequently, as described above at each node we first choose a team i whose break is to be fixed next. Then, we decide which breaks are infeasible for i considering the set of breaks already fixed for the corresponding subproblem. The tests to check infeasibility of a specific break are developed in Section 3.2. Further details of the branching scheme are outlined in Section 3.3.

Of course, setting the break for each team cannot guarantee the resulting LP subproblem to have an integer optimal solution. In order to emphasize the impact of our branching scheme we consider the IP corresponding to the subproblem in such a case and solve it using Cplex. Our experience with basic problems, see Briskorn and Drexl [5], tells us that only a very small amount of IPs have to be solved.

3.2 Identifying Infeasible Breaks

In the following we will call a break which has not been declared infeasible yet a potemtial break. Then, a child node i created for each potential break. The tests for infeasibility can be separated into four classes:

- The anonymous tests proposed in Briskorn and Drexl [5] to check whether a break is infeasible even if we do not consider individual requirements arising for teams. Thus, if we decide that a specific break is infeasible in general, then this break is infeasible for each team. The basic necessary conditions checked here are given above. For details about an efficient implementation we refer to Briskorn and Drexl [5].
- We develop static individual tests deciding whether a specific break is infeasible for team i given stadium unavailabilities. We describe the reduction of the set of potential breaks for each team in Section 3.2.1.
- We propose dynamic individual tests deciding whether a specific break is infeasible for team i given fixed matches, regions' capacities, and a set of already fixed breaks and the corresponding teams. We describe the reduction of the set of potential breaks for each team in Section 3.2.2.
- Dynamic tests for consistency of the sets of potential breaks are outlined. We develop an efficient technique checking a necessary condition for consistency of the set of potential breaks in a certain subproblem in Section 3.2.3.

3.2.1 Considering Stadium Unavailability

Stadium unavailability for team i in period p trivially means that team i can have a home break neither in p nor in $p+1$. Moreover, since there is exactly one break per team (including breaks in the first period) we can determine which breaks would lead to a match of i at home in p. Breaks unconditionally leading to a match of i at home in p are home breaks in a period in set

$$P^h = \{p' \mid p' \leq p, p - p' \bmod 2 = 0\} \cup \{p' \mid p' > p, p' - p \bmod 2 = 1\}$$

and away breaks in a period in set

$$P^a = \{p' \mid p' \leq p, p - p' \bmod 2 = 1\} \cup \{p' \mid p' > p, p' - p \bmod 2 = 0\}$$

Thus, home breaks in P^h and away breaks in P^a are infeasible for i. If team i has to play at home in period p we can determine the breaks being infeasible by exchanging P^h and P^a.

If there is more than one period where stadium unavailability restricts the venue a team can play at, then we can apply the same technique mentioned above for each such period. However, in the following we present a more efficient way. We look at periods in a circular fashion, that is after the last period $n-1$ follows period 1. Let $P^s = \{p_1, \ldots, p_k\}$, $k \geq 2$, be the set of periods where team i cannot play at home or away. Then there is one k', $1 \leq k' \leq k$, such that the break of i must occur in periods $p_{k'} + 1$ to $p_{k'+1}$ where $k + 1 = 1$. This is obvious from the fact that there is exactly one break per team and the intervals derived from P^s are disjoint. We can identify the interval by the following rules.

Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Venue	–	H	–	–	–	H	–	–	–	A	–	–	–	–	–	A	–
Breaks for 2	A	H	H	A	H	A	H	A	H	A	H	A	H	A	H	A	H
Breaks for 6	A	H	A	H	A	H	H	A	H	A	H	A	H	A	H	A	H
Breaks for 10	H	A	H	A	H	A	H	A	H	A	A	H	A	H	A	H	A
Breaks for 16	H	A	H	A	H	A	H	A	H	A	H	A	H	A	H	A	A
Pot. Breaks	–	–	–	–	–	–	H	A	H	A	–	–	–	–	–	–	–

Table 3: Feasible Break for given Stadium Unavailability

1. If $p_{k'+1} - p_{k'}$ is even and the venues of i in $p_{k'}$ and $p_{k'+1}$ are different, then there must be a break of i in periods $p_{k'} + 1$ to $p_{k'+1}$.
2. If $p_{k'+1} - p_{k'}$ is odd and the venues of i in $p_{k'}$ and $p_{k'+1}$ are identical, then there must be a break of i in periods $p_{k'} + 1$ to $p_{k'+1}$.
3. If none of the above, then i cannot have a break in periods $p_{k'} + 1$ to $p_{k'+1}$.

Rules 1 and 2 are obvious since the venues in $p_{k'}$ and $p_{k'+1}$ do not allow a HAP that is strictly alternating between $p_{k'}$ and $p_{k'+1}$. Rule 3 is due to the fact that no team has more than one break. If there is one break in periods $p_{k'} + 1$ to $p_{k'+1}$, then there must be a second break in periods $p_{k'} + 1$ to $p_{k'+1}$ due to the venues in $p_{k'}$ and $p_{k'+1}$.

Table 3 lines out the application of the rules mentioned above. The second line "Venue" gives four fixed venues for team i in periods 2, 6, 10, and 16. Each of the following lines gives the possible breaks regarding one of the fixed venues. Here "Breaks for p" refers to the fixed venue in period p. This line gives the venues of the break of i if its break occurs in the period corresponding to the column taking into account the venue of i in p. Thus, the entry for period 4 in line "Breaks for 10" can be read as if i has its break in period 4, then it is an away break since i plays away in period 10.

For each fixed venue of team i we have the venue of the break fixed depending on the period of the break. Hence, if two fixed venues of a break in period p are conflicting, then p cannot be a break period of i. Consider period 3 for example. From the facts that team i plays at home in 2 and 6 we conclude that a break of i in 3 can only occur at home and away, respectively. Obviously, this is a contradiction and, therefore, the assumption that the break occurs in 3 must be wrong. Thus, we can drop both breaks in 3 from the set of potential breaks of i.

The last line of Table 3 gives the venues for each period where all lines coincide. These symbolize the only possible breaks regarding the fixed venues of i. A column without entry represents a period where both breaks are infeasible for i.

Now consider the application of rules 1 to 3 to the example given in Table 3. We have $k = 4$ and $p_1 = 2$, $p_2 = 6$, $p_3 = 10$, and $p_4 = 16$. Rule 2 applies to $k' = 2$ while for each $k' \neq 2$ Rule 3 applies. This is reflected by the last line in Table 3.

3.2.2 Considering Fixed Matches and Regions' Capacities

In the following we develop a mechanism in order to identify infeasible breaks regarding fixed matches and regions' capacities. Again, we restrict the set of potential breaks of each team

Period	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Break	–	–	1	–	–	–	2	–	–	–	3	–	–	–	–	–	–
HAP 1	A	H	H	A	H	A	**H**	**A**	**H**	**A**	H	A	H	A	H	A	H
HAP 2	A	H	**A**	H	A	H	**H**	**A**	**H**	**A**	H	A	H	A	H	A	H
HAP 3	H	A	H	A	H	A	**H**	**A**	**H**	**A**	A	H	**A**	H	A	H	A
Venues 4	–	–	–	–	–	–	**A**	**H**	**A**	**H**	–	–	–	–	–	–	–
Venues 5	–	–	**H**	–	–	–	–	–	–	–	–	–	**H**	–	–	–	–
Breaks 4	A	H	A	H	A	H	A	–	–	–	H	A	H	A	H	A	H
Breaks 5	H	A	H	–	–	–	–	–	–	–	–	–	–	H	A	H	A

Table 4: Feasible Breaks for given Fixed Matches and Regions' Capacities

by identifying periods where a team i cannot play at home and away, that is periods where the venue of i can be considered fixed. Both, regions' capacities as well as fixed matches, can only lead to a fixed venue for team i under special circumstances.

1. If team j is fixed to play against team i in period p and, furthermore, j is fixed to play at home and away in p, then i must play away and at home, respectively, in p.
2. If $i \in R$ and there is a set $T' \subset R$, $|T'| = C_R$, of teams such that each team in T' plays at home (away) in p, then i (as well as all other teams in $R \setminus T'$) must play away (at home) in p.

While chances are low that one of these rules is applicable for the initial problem chances get higher during the branching process since fixing breaks implies fixing venues for teams. Hence, after fixing a break for team j we check whether $(\min\{i,j\}, \max\{i,j\}) \in F$ for any team $i \neq j$. If so Rule 1 applies for i. Furthermore, for each $R \ni j$ we update the remaining home capacity and the remaining away capacity of R in each period p, that is the number of teams of R being allowed to play at home and away, respectively, in addition to those teams of R having a fixed venue in p. If the remaining home capacity or the remaining away capacity is reduced to zero for any period p, then Rule 2 applies for region R and period p and, therefore, for each team $i \in R$, $i \neq j$. If one of these rules is applicable and tightens the restrictions for the venue where i competes in any period p, then we can consider this restriction as stadium unavailability in p and proceed as in Section 3.2.1.

Table 4 illustrates the procedure specified above. Let us assume we already fixed breaks for three teams which results into three fixed HAPs. Then, the second line gives for each period p the team that has a break already fixed in p. The resulting HAPs are given in lines "HAP 1", "HAP 2", and "HAP 3". We refer to the corresponding teams as Teams 1, 2, and 3 in the following. Let us further assume that teams 1, 2, and 3 belong to a region R having capacity $C_R = 3$. Then, teams 1, 2, and 3 use up the region's capacity in periods 7 to 10. This means, that a fourth team (team 4 in the following) of R has to play away in periods 7 and 9 and has to play at home in periods 8 and 10. These fixed venues are outlined in line "Venue 4". Employing the technique outline in Section 3.2.1 we find that only breaks according to line "Breaks 4" are potential for team 4.

Now suppose that a fifth team (team 5 that does not necessarily belong to R) is fixed to play against teams 2 and 3 in periods 3 and 13, respectively. Then, team 5 has to play at home in

periods 3 and 13 since its opponent plays away. This is symbolized in line "Venues 5". Again, employing the technique outline in Section 3.2.1 we find that only breaks according to line "Breaks 5" are possible for team 5.

3.2.3 Finding Pairs of Teams

In this Section we develop a procedure to further reduce the set of potential breaks. The focus here is on the combination of sets of potential breaks. We employ a property of feasible HAP sets having the minimum number of breaks. Among others Miyashiro et al. [21] show that in such a HAP set there are exactly $n/2$ periods having two breaks. Obviously, two teams having their breaks in the same periods have complementary HAPs, that is in each period their venues are different from each other. Thus, in a feasible HAP set we have $n/2$ pairs of teams connected to each other by identical break periods and complementary HAPs.

For a given partial HAP set and a set of potential breaks for each team we check whether we can pair up teams such that both teams in each pair can have complementary HAPs. Note that we can see a fixed break for team i as the only potential break for i. Then, two teams i and j can form a pair if there is at least one period p such that i and j have potential breaks in p in different venues. Next, we show how to decide whether we can pair up all teams. We define an undirected graph $G = (V, E)$ as follows.

$$\begin{aligned} V &= T \\ E &= \{(i,j) \mid i,j \in T, i < j,\ i \text{ and } j \text{ can form a pair}\} \end{aligned}$$

Using the algorithm by Edmonds [18] we can find a maximum matching in G in polynomial time. If the maximum matching is a perfect one, that is it contains $n/2$ edges, then this matching represents a partition of T into pairs such that the teams in each pair can have complementary HAPs. If the maximum matching has less than $n/2$ edges, then the partial HAP set is infeasible. Obviously, the existence of a perfect matching is a necessary condition for the partial HAP set to be feasible. However, it is no sufficient one. This is due to the fact that we cannot consider the set of break periods resulting from the pairs of teams. For example, the actual pairs may require to set breaks in three consecutive periods. However, it is well known, see Briskorn and Drexl [4] for example, that in a feasible HAP set with minimum number of breaks there cannot be three consecutive break periods.

Period	1	2	3	4	5	6	7
Team 1	–	A	–	–	–	–	–
Team 2	–	–	H	–	–	–	–
Team 3	–	–	A	–	–	–	–
Team 4	–	–	–	A	**A**	–	–
Team 5	A	H	A	H	A	H	A
Team 6	H	A	H	–	–	–	–
Team 7	–	–	–	–	H	A	H
Team 8	A	H	–	–	–	–	–

Table 5: Feasible Breaks

In order to illustrate the procedure described above we provide Table 5 and Figure 1. In Table 5 a set of potential breaks is defined for each of 8 team. We assume that fixed breaks are projected on potential breaks as described above. On the left hand side of Figure 1 we provide the corresponding graph G. Since in our example there is no team having two potential breaks in at least one period we can state that edge (i, j) exists in G if and only if there is a period p such that i and j have different potential breaks in p. The bold lines in the graph on the left hand side represent a perfect matching in G. Thus, there is a matching having cardinality $n/2$ and, therefore, the set of potential breaks in Table 5 fulfills our necessary condition.

Next, let us assume that we branch on the break of team 4 and fix team 4 to have an away break in period 4 first. Then, the entry in bold lettering in Table 5 is obsolete and the graph on the right hand side of Figure 1 is the corresponding graph G. Now, there is no matching having cardinality of more than 3. Thus, there is no perfect matching and an away break in period 4 cannot be a feasible choice for team 4.

3.3 Strategies to Explore the Search Tree

3.3.1 Choice of Branching Candidate

The branching scheme prescribes to create a child node for each potential break of a specific team. Consequently, branching candidates correspond to teams. In the following we develop strategies to choose branching candidates.

The first one is derived from Briskorn and Drexl [5]. Considering the LP relaxation of (1) to (12) we define a fractional break value

$$\overline{br}_{i,p} = |1 - \sum_{j \in T \setminus \{i\}} (x_{i,j,p-1} + x_{i,j,p})|$$

where $x_{i,j,0}$ means $x_{i,j,n-1}$ for each team $i \in T$ and period $p \in P$. Then, 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]. In the LP relaxation of (1) to (12) it is possible to have zero breaks and, thus, there may be teams without any break. On the other hand, there may be teams having more than one break. Since this severely contradicts the structure of a feasible IP solution we choose team

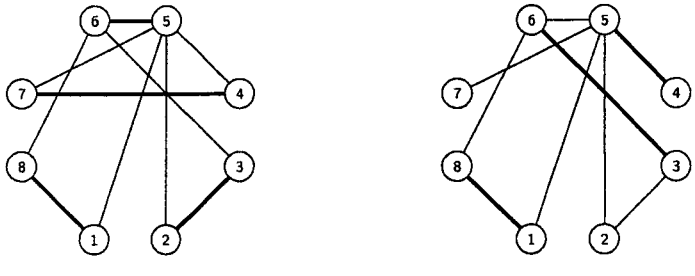

Figure 1: Graphs for Maximum Cardinality Matching Problem

$$i' = \arg\max_i \left\{ \left| \sum_{p \in P} \overline{br}_{i,p} - 1 \right| \right\}$$

as branching team if $\sum_{p \in P} \overline{br}_{i',p} \neq 1$. Otherwise (if $\sum_{p \in P} \overline{br}_{i,p} = 1$ for each $i \in T$), we choose team

$$i^* = \arg\max_i \left\{ \max_p \{ \overline{br}_{i,p} \mid \overline{br}_{i,p} < 1 \} \right\}$$

with largest fractional break value. The motivation for choosing i^* 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.

The second strategy to choose branching candidates aims at increasing probability to find feasible solutions in an early stage of the B&B algorithm. We choose the team having the smallest number of potential breaks. In some sense this team can be seen as a bottleneck and not considering this team now will further decrease probability to find a feasible break in future stages.

3.3.2 Node Order Strategy

We observe that "Breadth First Search" performs significantly better than "Depth First Search" for instances with up to 8 teams. For larger instances, however, the search tree grows to large and the administrative overhead equalizes this advantage or – even worse – the algorithm aborts due to lack of memory.

Therefore, we employ 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 a key value derived from the father node's optimal LP solution. We normalize the father node's solution value dividing it by the lower bound given by the root node's optimal LP solution and obtain v_f as normalized value. Additionally, we calculate the fraction x_f of match variables having binary values in the father node's optimal solution. Then, we sort the node list in non-decreasing order of $v_f - w \cdot x_f$ 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 $w = 1.5$ turned out to be a suitable value.

Furthermore, we need a tie breaker for child nodes of the same father node since these nodes are rated equal according to the mechanism proposed above. For a specific child node and, thus, for a specific break for team i we consider the average value of those matches that are possible for i if this break is fixed. Then, we sort child nodes in non-decreasing order of this value.

4 Computational Results

We carried out our computational study using a 3.8 GHz Pentium 4 machine with 3 GBs of RAM running Microsoft Windows Server 2003. We compared our approach to Ilog Cplex 10.1 with default settings referred to as "CPLEX" in the following. We employed the strategy using

fractional break values in order to choose a branching candidate and the strategy described in Section 3.3.2 as node order strategy. For solving the maximum cardinality matching problem introduced in Section 3.2.3 we employed the Boost Graph Library (see Siek et al. [26]). Here, the implementation of Edmonds' algorithm follows Tarjan [27]. We compiled our algorithm coded in C++ using Microsoft Visual Studio 2005.

For each number of teams between 6 and 12 we have 9 classes of 20 instances each. In each class costs are randomly drawn from integer values between 0 and 20. In the following we specify the characteristics of each class.

1. In class "MCSRRTns" the actual problem is defined by (1) to (7) and (11) as well as (12) with $F = \emptyset$. That is, we do not consider fixed matches, stadium availability, or regions. Then, instances are fully specified by n.

2. Class "MCSRRTs10" contains instances of problem MCSRRTs. For each team i and each period p it is randomly chosen whether i can play at home and away in p or not. Probability for a restriction of possible venues is 10%. If the possible venue is restricted, then it is randomly chosen whether the stadium is unavailable in the first or the second half of the tournament with equal probability.

3. Class "MCSRRTs20" is defined just as class "MCSRRTs10" except for the probability of restricted venues being 20%.

4. Class "MCSRRTf2" contains instances of problem MCSRRTf. We do randomly choose whether a specific match of i at home against j in p is fixed or not. Probability for a match to be fixed is 2%.

5. Class "MCSRRTf4" is defined just as class "MCSRRTf2" but probability for a match to be fixed is 4%.

6. Class "MCSRRTr1" contains instances of problem MCSRRTr. We do consider two regions of size $n/2$ each. The capacity of each region R is given as $C_R = \lceil |R| \rceil$.

7. Class "MCSRRTr2" is defined just as class "MCSRRTr1" except for the sizes of the regions. Sizes of regions in "MCSRRTr2" are $n/2 + 1$ and $n/2 - 1$.

8. Class "MCSRRT1" contains instances of problem MCSRRT. Parameters are chosen according to "MCSRRTs10", "MCSRRTf2", and "MCSRRTr1".

9. Class "MCSRRT2" contains instances of problem MCSRRT as well. Parameters are chosen according to "MCSRRTs20", "MCSRRTf4", and "MCSRRTr2".

Results of our computational study, that is average run times in seconds for each problem size, instance class, and approach in seconds, are outline in Table 6. For each size of instances we distinguish between finding the optimal solution ("find opt") and proving its optimality ("prove opt"). For both tasks we present the run times needed to accomplish the corresponding task. There are classes of instances where Cplex could solve some but not all of the instances. The number of instances solved (a proven optimal solution has been found for) is outlined in brackets after the run times needed by Cplex. If no number is given, then all instances have been solved. In the column entitled "B&B*" we outline the average run times needed by B&B

for instances Cplex could solve. For our algorithm we additionally outline the number of IP subproblems being solved on average in column "# IPs".

We clearly see that the average run time needed by B&B is significantly lower than the one needed by Cplex for each problem size and instance class. The superiority is higher for larger problem sizes. Although the superiority is general we can identify instance classes where it is higher than in others. Whenever stadium unavailability is considered we obtain very pleasant run times even for problem sizes both approaches cannot handle if stadium unavailability is not an issue.

We can see that if Cplex did not solve all instances of a certain class and size, then tackling the solved instances with B&B leads to lower run times than those obtained using B&B for the whole class. Thus, it seems like both approaches behave similar as far relation of performance on different sets of instances is concerned. This impression can be reassured by comparing performances on different classes and sizes in Table 6.

By comparing the run times for MCSRRTns with those for other classes we have to take into account that B&B can make use of the symmetry between teams in MCSRRTns. Hence, the fact that average run times for MCSRRTs, MCSRRTf, and MCSRRT are lower is even more remarkable since we cannot use any symmetry here anymore. The only class of instances sticking out with regard to run times is MCSRRTr. However, taking into account missing symmetry we can say that the approach performs well in comparison to MCSRRTns.

Furthermore, comparing run times needed to find the optimal solution the superiority of our approach is confirmed. For each but one problem size and instance class B&B finds the optimal solution significantly faster on average than Cplex does. Taking a closer look at MCSRRTns with 10 we see that Cplex finds the optimal solution faster here. However, it is justified to say that this is an odd event since it happens for the only out of 20 instances that could be solved at all. Mostly, both phases – finding the optimal solution and proving its optimality after having found it – are done faster on average by our approach. Exceptions for this statement are MCSRRTf2 and MCSRRTr2 with 8 teams. Considering MCSRRTf2 for 8 team for example, finding the feasible solution is done using 1.1 seconds less of run time on average by our approach. However, proving the optimality is done using only 0.72 seconds less of run time, that is we loose 0.38 seconds in the second phase. This loss may be caused by Cplex using various techniques in order to tighten the lower bounds, adding cuts for example. Note that we do not use such techniques at all in order to emphasize the impact of the branching scheme.

Last but not least we see that the number of IP subproblems being solved on average per instance is small enough to be accepted. Hence, although our branching scheme is not complete in a sense that it can guarantee an integer optimal solution in each LP subproblem this case occurs very rarely. We would like to emphasize that not a single infeasible IP subproblem, that is an IP subproblem corresponding to an infeasible HAP set, has been solved. Consequently, we can afford to solve these subproblems as IPs in terms of run time.

5 Conclusions and Outlook

In the paper at hand we develope a branching scheme in order to solve the problem to find a cost minimal single RRT considering side constraints, namely minimum number of breaks, stadium unavailability, fixed matches, and regions' capacities. Branching is done by fixing

breaks for teams. Here, the focus is on finding out whether a specific break for a certain team can lead to a feasible HAP set. We develop efficient tests to identify infeasible breaks.

We show by means of a computational study that our approach is able to solve problem instances of up to 10 teams (12 for some variants of the problem) in significantly less run time than Cplex. Since the gap of run times of our approach and Cplex is widening with increasing number of teams we expect the superiority to hold for larger instances also. However, run times are growing unreasonably large for larger instances.

Although the branching scheme itself cannot guarantee integer solutions our results show that almost no LP subproblem without an optimal integer solution but without any branching candidate is left. Hence, it is justified to solve these rare subproblems as an IP.

Even though the branching scheme is able to provide a significant reduction of run times we can obtain optimal solutions only for very few real world sports leagues since common leagues sizes are $n = 18$ and $n = 20$ teams. There are two obvious ways to proceed in order to develop algorithms for these problem sizes. First, the bounding components of our B&B algorithm can be improved. As mentioned before, so far the only bounding is done by considering the lower bound given by the LP relaxation of (1) to (12) with fixed breaks. Here, cut generation techniques could help to tighten the lower bound. Second, we can refuse to search the whole B&B tree and finding a good solution in a well chosen part of the tree.

Another field of work is to employ the knowledge gained in the paper at hand about feasibility of partial HAP sets in other solution techniques, constraint programming approaches for example.

References

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

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

[3] D. Briskorn. Feasibility of home-away-pattern sets for round robin tournaments. *Operations Research Letters*, 36:283–284, 2008.

[4] D. Briskorn and A. Drexl. Branching Based on Home–Away–Pattern Sets. In K.-H. Waldmann and U. M. Stocker, editors, *Operations Research Proceedings 2006 - Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Karlsruhe, September 6th - 8th 2006*, pages 523–528. Springer, Berlin, Germany, 2007.

[5] D. Briskorn and A. Drexl. A branching scheme for finding cost-minimal round robin tournaments. *European Journal of Operational Research*, 197(1):68–76, 2009.

[6] D. Briskorn and A. Drexl. Integer Programming Models for Round Robin Tournaments. *Computers & Operations Research*, 36(3):837–852, 2009.

[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, 1988.

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

[15] F. Della Croce and D. Oliveri. Scheduling the Italian Football League: An ILP–Approach. *Computers & Operations Research*, 33:1963–1974, 2006.

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

[17] G. Durán, M. Guajardo, J. Miranda, D. Sauré, and A. Weintraub. Scheduling the Chilean Soccer League by Integer Programming. *Interfaces*, 37:539–552, 2007.

[18] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[19] G. Kendall. Scheduling English Football Fixtures Over Holiday Periods. *Journal of the Operational Research Society*, 59:743–755, 2008.

[20] S. Knust. Classification of literature on sports scheduling, 2009. URL `http://www.inf.uos.de/knust/sportlit_class/`. (January 29^{th}, 2009).

[21] 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, *The 4th International Conference on the Practice and Theory of Automated Timetabling, Selected Revised Papers*, Lecture Notes in Computer Science 2740, pages 78–99. Springer, Berlin, Germany, 2003.

[22] R. V. Rasmussen. Scheduling a Triple Round Robin Tournament for the Best Danish Soccer League. *European Journal of Operational Research*, 185:795–810, 2008.

[23] R. V. Rasmussen and M. A. Trick. Round Robin Scheduling – a survey. *European Journal of Operational Research*, 188:617–636, 2008.

[24] C. C. Ribeiro and S. Urrutia. Scheduling the Brazilian soccer tournament with fairness and broadcast objectives. Lecture Notes in Computer Science 3867, pages 149–159, 2007.

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

[26] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual.* Addison-Wesley Professional, 2001.

[27] R. E. Tarjan. *Data Structures and Network Algorithms.* Society for Industrial and Applied Mathematics, 1983.

Size	6						8					
Goal	prove opt				find opt		prove opt				find opt	
Approach	Cplex	B&B	B&B*	# IPs	Cplex	B&B	Cplex	B&B	B&B*	# IPs	Cplex	B&B
MCSRRTns	0.20	0.12	—	0.15	—	—	27.75	18.16	—	0.40	16.54	10.39
MCSRRTs10	0.14	0.06	—	0.05	—	—	14.57	3.08	—	0.85	11.84	1.13
MCSRRTs20	0.11	0.03	—	0.30	—	—	5.09	0.68	—	0.25	3.22	0.37
MCSRRTf2	0.11	0.06	—	0.05	—	—	5.56	4.84	—	0.05	4.12	3.02
MCSRRTf4	0.10	0.06	—	0.00	—	—	5.93	4.37	—	0.00	4.72	3.18
MCSRRTr1	0.24	0.14	—	0.05	—	—	37.56	14.95	—	0.55	30.87	10.42
MCSRRTr2	0.29	0.15	—	0.10	—	—	37.05	22.88	—	0.70	30.31	12.66
MCSRRT1	0.10	0.04	—	0.10	—	—	3.08	0.72	—	0.20	1.25	0.35
MCSRRT2	0.06	0.02	—	0.00	—	—	0.94	0.25	—	0.15	0.21	0.17

Size	10						12					
Goal	prove opt				find opt		prove opt				find opt	
Approach	Cplex	B&B	B&B*	# IPs	Cplex	B&B	Cplex	B&B	B&B*	# IPs	Cplex	B&B
MCSRRTns	14708.80 (01)	7.116.18	3981.45	4.15	3024.00	3126.50	—	—	—	—	—	—
MCSRRTs10	21511.63 (08)	333.08	159.13	3.25	18859.5	140.5	—	18437.92 (14)	—	7.71	—	10155.36
MCSRRTs20	9640.58 (19)	17.87	15.25	2.80	6652.00	6.45	—	160.63 (19)	—	4.58	—	112.36
MCSRRTf2	22213.17 (16)	665.56	579.55	1.25	18165.00	302.40	—	—	—	—	—	—
MCSRRTf4	12308.50 (17)	317.95	244.74	0.55	6993.65	160.65	—	—	—	—	—	—
MCSRRTr1	—	7641.75	—	3.40	—	5541.34	—	—	—	—	—	—
MCSRRTr2	—	3699.85	—	3.55	—	2388.23	—	—	—	—	—	—
MCSRRT1	7559.11 (19)	68.90	58.21	0.85	5370.63	39.74	—	141.16 (20)	—	0.75	—	85.49
MCSRRT2	310.95 (20)	1.60	—	0.35	254.20	0.95	—	56.75 (20)	—	2.10	—	26.86

Table 6: Computational results