Integer Programming Models for Round Robin Tournaments

Jasper van Doornmalen, Christopher Hojny*, Roel Lambers, and Frits C.R. Spieksma

Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600 MB Eindhoven, The Netherlands {m.j.v.doornmalen, c.hojny, r.lambers, f.c.r.spieksma}@tue.nl

Abstract

Round robin tournaments are omnipresent in sport competitions and beyond. We propose two new integer programming formulations for scheduling a round robin tournament, one of which we call *the matching formulation*. We analytically compare their linear relaxations with the linear relaxation of a well-known traditional formulation. We find that the matching formulation is stronger than the other formulations, while its LP relaxation is still being solvable in polynomial time. In addition, we provide an exponentially sized class of valid inequalities for the matching formulation. Complementing our theoretical assessment of the strength of the different formulations, we also experimentally show that the matching formulation is superior on a broad set of instances. Finally, we describe a branch-and-price algorithm for finding round robin tournaments that is based on the matching formulation.

Keywords: Integer Programming, OR in sports, Cutting planes, Branch-and-price

1 Introduction

Integer programming continues to be a very popular way to obtain a schedule for a round robin tournament. The ability to straightforwardly model such a tournament, and next solve the resulting formulation using an integer programming solver, greatly facilitates practitioners. Moreover, it is usually possible to add all kinds of specific local constraints to the formulation that help addressing particular challenges. We substantiate this claim of the widespread use of integer programming by mentioning some of the works that use integer programming to arrive at a schedule for a round robin tournament. Indeed, from the literature, it is clear that for national football leagues (which are predominantly organized according to a so-called double round robin format), integer programming-based techniques are used extensively to find schedules. Without claiming to be exhaustive we mention Alarcón et al. [2], Della Croce and Oliveri [9], Durán et al [11, 12, 10], Goossens and Spieksma [17], Rasmussen [27], Recalde et al. [29], Ribeiro and Urrutia [30]. Other sport competitions that are organized in a round robin fashion (or a format close to a round robin) have also received ample attention: we mention Cocchi et al. [8] and Raknes and Pettersen [26] who use integer programming for scheduling volleyball leagues, Fleurent and Ferland [16] who use integer programming for scheduling a hockey league, Kim [21] and Bouzarth et al. [4] for baseball leagues, Kostuk and Willoughby [23] for Canadian football, Nemhauser and Trick [25] and Westphal [36] for basketball leagues. Further, there has been work on studying properties of the traditional formulation, among others, by Trick [33] and Briskorn and Drexl [5]. Well-known surveys are given by Rasmussen and Trick [28], Kendall et al. [20] and Goossens and Spieksma [18]; we also refer to Knust [22], who maintains an elaborate classification of literature on sports scheduling. More recently, the international timetabling competition [35] featured a

*Corresponding author

round robin sports timetabling problem, and most of the submissions for this competition used integer programming in some way to obtain a good schedule.

All this shows that integer programming is one of the most preferred ways to find schedules for competitions organized via a round robin format.

In this paper, we aim to take a fresh look at the problem of finding an optimal schedule for round robin tournaments using integer programming techniques. Depending upon how often a pair of teams is required to meet, different variations of a round robin tournament arise: in case each pair of teams meets once, the resulting format is called a Single Round Robin, in case each pair of teams is required to meet twice, we refer to the resulting variation as a Double Round Robin. These formats are the ones that occur most in practice; in general we speak of a k-Round Robin to describe the situation where each pair of teams is required to meet k times.

We have organized the paper as follows. In Section 2, we precisely define the problem corresponding to the Single Round Robin tournament, and we present three integer programming formulations for it. We call them the traditional formulation (Section 2.1), the matching formulation (Section 2.2), and the permutation formulation (Section 2.3); the latter two formulations are, to the best of our knowledge, new. We show that their linear relaxations can be solved in polynomial time. We prove in Section 3 that the matching formulation is stronger than the other formulations. In Section 4 we provide a class of valid inequalities for the matching formulation. We show in Section 5 how our results extend to the k-Round Robin tournament. In Section 6, we generate instances of our problem with two goals in mind: (i) to experimentally assess the quality of the bounds found by our models (Section 6.2), and (ii) to report on the performance of a branch-and-price algorithm (Section 6.3). We conclude in Section 7.

2 Problem definition and formulations

In this section, we provide a formal definition of our problem and introduce the necessary terminology and notation. We start by describing the so-called Single Round Robin (SRR) tournament, where every pair of teams has to meet exactly once, and we return to the general version of the problem, where every pair of teams has to meet k times ($k \geq 1$), in Section 5.

Throughout the entire paper, we assume that n is an even integer that denotes the number of teams; for reasons of convenience we assume $n \geq 4$. We denote the set of all teams by T. A *match* is a set consisting of two distinct teams and the set of all *matches* is denoted by $\mathcal{M}$, in formulae, $\mathcal{M} = \{m = \{i, j\} : i, j \in T,\ i \neq j\}$. We denote, for each $i \in T$, by $\mathcal{M}_i = \{\{i, j\} : j \in T \setminus \{i\}\}$ the set of matches played by team i. As we assume in this section that every pair of teams meets once, and as n is even, the matches can be organized in $n - 1$ rounds, which we denote by R; hence, we deal in this section with a *compact* single round robin tournament.

Prepared with this terminology and notation, we are able to provide a formal definition of the SRR problem.

Problem 1. (SRR) Given an even number $n \geq 4$ of teams with corresponding matches $\mathcal{M}$, a set of $n - 1$ rounds R, as well as an integral cost $c_{m,r}$ for every match $m \in \mathcal{M}$ and round $r \in R$, the *single round robin (SRR)* problem is to find an assignment $\mathcal{A} \subseteq \mathcal{M} \times R$ of matches to rounds that minimizes the cost $\sum_{(m,r)\in\mathcal{A}} c_{m,r}$ such that every team plays a single match per round and each match is played in some round.

Since the SRR problem is NP-hard (see Easton [13], Briskorn et al. [6], and Van Bulck and Goossens [34]), there does not exist a polynomial time algorithm to find an optimal assignment unless P = NP. For this reason, several researchers have investigated integer programming (IP) techniques for finding an optimal assignment of matches to rounds. We follow this line of research and discuss three different IP formulations for the SRR problem: a traditional formulation with polynomially many variables and constraints (Section 2.1) as well as two formulations that involve exponentially many variables (Sections 2.2 and 2.3). To the best of our knowledge, the latter models have not been discussed in the literature before.

2.1 The traditional formulation

The *traditional formulation* of the SRR problem has been discussed, among others, by Trick [33] and Briskorn and Drexl [5]. To model an assignment of matches to rounds, this formulation introduces, for every match $m \in \mathcal{M}$ and round $r \in R$, a binary decision variable $x_{m,r}$ to model whether match m is played at round r ($x_{m,r} = 1$) or not ($x_{m,r} = 0$). With these variables, problem SRR can be modeled as:

$$\min \sum_{m\in\mathcal{M}} \sum_{r\in R} c_{m,r} x_{m,r} \tag{T1}$$

$$\sum_{r\in R} x_{m,r} = 1, \qquad m \in \mathcal{M}, \tag{T2}$$

$$\sum_{m\in\mathcal{M}_i} x_{m,r} = 1, \qquad i \in T, r \in R, \tag{T3}$$

$$x_{m,r} \in \{0,1\}, \qquad m \in \mathcal{M}, r \in R. \tag{T4}$$

Constraints (T2) ensure that each pair of teams meets once, and Constraints (T3) imply that each team plays in each round. This model has $O(n^2)$ constraints and $O(n^3)$ variables. Note that Constraints (T4) can be replaced by $x_{m,r} \in \mathbb{Z}_+$ as the upper bound $x_{m,r} \leq 1$ is implicitly imposed via Constraints (T2) and non-negativity of variables. The linear programming relaxation of (T) arises when we replace (T4) by $x_{m,r} \geq 0$; given an instance I of SRR, we denote the resulting value by $v^{LP}_{tra}(I)$.

2.2 The matching formulation

Consider the complete graph that results when associating a node to each team, say $K_n = (T, \mathcal{M})$. Clearly, a single round of a feasible schedule can be seen as a perfect matching in this graph. This observation allows us to build a matching based formulation by introducing a binary variable for every perfect matching in K_n; we denote the set of all perfect matchings in K_n by $\mathfrak{M}$.

We employ a binary variable $y_{M,r}$ for each perfect matching $M \in \mathfrak{M}$ and round $r \in R$. If $y_{M,r} = 1$, the model prescribes that matching M is used for the schedule of round r, whereas $y_{M,r} = 0$ encodes that a different schedule is used. To be able to represent the cost of round $r \in R$, the total cost of all matches in M is denoted by $d_{M,r} \coloneqq \sum_{m\in M} c_{m,r}$, which leads to the model

$$\min \sum_{M\in\mathfrak{M}} \sum_{r\in R} d_{M,r} y_{M,r} \tag{M1}$$

$$\sum_{M\in\mathfrak{M}} y_{M,r} = 1, \qquad r \in R, \tag{M2}$$

$$\sum_{\substack{M\in\mathfrak{M}:\\ m\in M}} \sum_{r\in R} y_{M,r} = 1, \qquad m \in \mathcal{M}, \tag{M3}$$

$$y_{M,r} \in \{0,1\}, \qquad M \in \mathfrak{M}, r \in R. \tag{M4}$$

Constraints (M2) ensure that a matching is selected in each round, while Constraints (M3) enforce that each pair of teams meets in some round. Similarly to the traditional formulation, we can replace (M4) by $y_{M,r} \in \mathbb{Z}_+$. In this way, the linear programming relaxation of (M) arises when replacing (M4) by $y_{M,r} \geq 0$; given an instance I of SRR, the resulting value is denoted by $v^{LP}_{mat}(I)$. Notice that this formulation uses an exponential number of variables, as the number of matchings grows exponentially in n. Thus, a relevant question is whether we can find v^{LP}_{mat} in polynomial time. The following observation shows that it can be answered affirmatively.

Lemma 2.1. *The LP relaxation of the matching formulation* (M) *can be solved in polynomial time.*

Proof. Due to the celebrated result by Grötschel et al. [19], it is sufficient to show that the separation problem for the constraints of the dual of the linear relaxation of Model (M) can be solved in polynomial time. To avoid an exponential number of variables in the dual, we replace Constraint (M4) by $y_{M,r} \in \mathbb{Z}_+$ as explained above. Then, by introducing dual variables α_r, $r \in R$, corresponding to Constraints (M2) and β_m, $m \in \mathcal{M}$, corresponding to Constraints (M3), the constraints of the dual of the LP relaxation of Model (M) are:

$$\alpha_r + \sum_{m \in M} \beta_m \leq d_{M,r}, \qquad\qquad M \in \mathfrak{M}, r \in R.$$

Given values for the dual variables, say $(\bar{\alpha}, \bar{\beta})$, the separation problem is to decide whether it satisfies all dual constraints. For fixed $r \in R$, we show that this problem can be solved in polynomial time. Thus, the assertion follows as there are only $O(n)$ rounds.

Indeed, if $r \in R$ is fixed, the problem reduces to check whether there exists a matching $M \in \mathfrak{M}$ such that

$$\bar{\alpha}_r + \sum_{m \in M} \bar{\beta}_m > d_{M,r} = \sum_{m \in M} c_{m,r} \quad \Leftrightarrow \quad \sum_{m \in M} (\bar{\beta}_m - c_{m,r}) > -\bar{\alpha}_r.$$

The latter inequality asks whether there exists a perfect matching of teams with weight greater than $-\bar{\alpha}_r$, where an edge m between two teams is assigned weight $(\bar{\beta}_m - c_{m,r})$. This problem can be solved in polynomial time by Edmonds' blossom algorithm [14, 15], which concludes the proof. □

2.3 The permutation formulation

Instead of fixing the schedule of a round, the *permutation formulation* fixes, for a given team, the order of the teams against which the given team plays its successive matches. That is, it introduces a variable for each team i and each permutation of $T \setminus \{i\}$. We denote the set of all such permutations by Π^{-i}. Moreover, for a team $j \in T$ and round $r \in R$, denote the set of all permutations where j occurs at position r in the permutation by $\Pi^{-i}_{j,r}$. Permutations from $\Pi^{-i}_{j,r}$ thus encode that team i plays against team j on round r. For a permutation $\pi \in \Pi^{-i}$ and round $r \in R$, we refer to the opponent of team i at round r as $\pi_r \in T \setminus \{i\}$. The cost of a schedule encoded via permutations Π^{-i} for a team $i \in T$ is then given by $e_{i,\pi} := \sum_{r \in R} c_{\{i,\pi_r\},r}$. Using binary variables $z_{i,\pi}$, where $i \in T$ and $\pi \in \Pi^{-i}$, that encode whether i plays against its opponents in order π ($z_{i,\pi} = 1$) or not ($z_{i,\pi} = 0$), the permutation formulation is

$$\min \frac{1}{2} \sum_{i \in T} \sum_{\pi \in \Pi^{-i}} e_{i,\pi} z_{i,\pi} \tag{P1}$$

$$\sum_{\pi \in \Pi^{-i}} z_{i,\pi} = 1, \qquad i \in T, \tag{P2}$$

$$\sum_{\pi \in \Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi \in \Pi^{-j}_{i,r}} z_{j,\pi}, \qquad \{i,j\} \in \mathcal{M}, r \in R, \tag{P3}$$

$$z_{i,\pi} \in \{0,1\}, \qquad i \in T, \pi \in \Pi^{-i}. \tag{P4}$$

Constraints (P2) ensure that a permutation is selected for each team, while Constraints (P3) enforce that, given a round and a pair of teams, these teams meet in that round, or they do not meet in that round. Due to rescaling the objective by $\frac{1}{2}$, we find the cost of an optimal SRR schedule. Moreover, we can again replace Constraint (P4) by $z_{i,\pi} \in \mathbb{Z}_+$. The linear programming relaxation of (P) then arises when replacing Constraints (P4) by $z_{i,\pi} \geq 0$; given an instance I of SRR, we denote the resulting value by $v^{LP}_{per}(I)$.

Since this model has $n!$ variables, we again investigate whether its LP relaxation can be solved efficiently.

Lemma 2.2. *The LP relaxation of the permutation formulation* (P) *can be solved in polynomial time.*

Proof. As in the proof of Lemma 2.1, it is sufficient to show that the separation problem corresponding to the constraints of the dual of the relaxation of the permutation formulation can be solved in polynomial time. Again, to avoid exponentially many variables in the dual, we replace (P4) by $z_{i,\pi} \in \mathbb{Z}_+$. We introduce dual variables α_i for each constraint of type (P2) and $\beta_{\{i,j\},r}$ for each constraint of type (P3). To normalize Constraint (P3), we assume it to be given by $\sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} - \sum_{\pi\in\Pi^{-j}_{i,r}} z_{j,\pi} = 0$ with $i < j$. Then, the dual constraints are given by

$$\alpha_i + \sum_{\substack{r\in R:\\ i<\pi_r}} \beta_{\{i,\pi_r\},r} - \sum_{\substack{r\in R:\\ i>\pi_r}} \beta_{\{i,\pi_r\},r} \leq \frac{1}{2}e_{i,\pi} \qquad\qquad i\in T, \pi\in\Pi^{-i}.$$

If $i \in T$ is fixed, the separation problem for dual values $(\bar{\alpha}, \bar{\beta})$ is to decide whether there exists a permutation $\pi \in \Pi^{-i}$ such that

$$\sum_{\substack{r\in R:\\ i<\pi_r}} \bar{\beta}_{\{i,\pi_r\},r} - \sum_{\substack{r\in R:\\ i>\pi_r}} \bar{\beta}_{\{i,\pi_r\},r} - \frac{1}{2}\sum_{r\in R} c_{\{i,\pi_r\},r} > -\bar{\alpha}_i$$

due to the definition of $e_{i,\pi}$. To answer this question, it is sufficient to find a permutation maximizing the left-hand side expression. Such a permutation can be found by computing a maximum weight perfect matching in the complete bipartite graph with node bipartition $(T \setminus \{i\}) \cup R$ and edge weights defined for each $j \in T \setminus \{i\}$ and $r \in R$ by

$$w_{j,r} = \begin{cases} -\frac{1}{2}c_{\{i,j\},r} + \bar{\beta}_{\{i,j\},r}, & \text{if } i < j,\\ -\frac{1}{2}c_{\{i,j\},r} - \bar{\beta}_{\{i,j\},r}, & \text{otherwise.}\end{cases}$$

Since this problem can be solved in polynomial time, the assertion follows by solving this problem for each of the n teams. □

3 Comparing the strength of the different formulations

In the previous section, we have introduced three different models for finding an optimal schedule for problem SRR. While the traditional formulation contains both polynomially many variables and constraints, the matching and permutation formulation make use of an exponential number of variables. The aim of this section is to investigate whether the increase in the number of variables in comparison with the traditional formulation leads to a stronger formulation. We measure the strength of a formulation based on the value of its LP relaxation, where a higher value of the LP relaxation indicates a stronger formulation as the LP relaxation's value is closer to the optimum value of the integer program, as encapsulated by the following definitions.

Definition 3.1. Let f and g be mixed-integer programming formulations of the SRR problem and denote by $v^{\mathrm{LP}}_f(I)$ and $v^{\mathrm{LP}}_g(I)$ the value of the respective LP relaxations for an instance I of SRR.

- We say that f and g are *relaxation-equivalent* if, for each instance I of problem SRR, the value of the linear programming relaxations are equal, i.e., $v^{\mathrm{LP}}_f(I) = v^{\mathrm{LP}}_g(I)$.
- We say that f is stronger than (or dominates) g if (i) for each instance I of problem SRR, $v^{\mathrm{LP}}_f(I) \geq v^{\mathrm{LP}}_g(I)$, and (ii) there exists an instance I of problem SRR for which $v^{\mathrm{LP}}_f(I) > v^{\mathrm{LP}}_g(I)$.

We now proceed by formally comparing the strength of the formulations from Section 2 using the terminology of these definitions. We state our results using three lemmata, and summarize all our results in Theorem 3.5.

First, we show that the traditional and permutation formulation have equivalent LP relaxations.

Lemma 3.2. *The permutation formulation* (P) *is relaxation-equivalent to the traditional formulation* (T).

Proof. To prove this lemma, we show that there is a one-to-one correspondence of feasible solutions of the traditional formulation's and the permutation formulation's LP relaxations that preserves the objective value. First, we construct a solution of the LP relaxation of the traditional formulation from a solution z of the LP relaxation of the permutation formulation. To this end, define for each $\{i, j\} \in \mathcal{M}$ and $r \in R$ a solution $x \in \mathbb{R}^{\mathcal{M}\times R}$ via $x_{\{i,j\},r} = \sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi}$. Note that x is non-negative as all z-variables are non-negative. Moreover, it is well-defined as $\sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi\in\Pi^{-j}_{i,r}} z_{j,\pi}$ due to (P3). Finally, all constraints of type (T2) and (T3) are satisfied since

$$\text{for each } m\in\mathcal{M}: \qquad \sum_{r\in R} x_{\{i,j\},r} = \sum_{r\in R}\sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi\in\bigcup_{r\in R}\Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi\in\Pi^{-i}} z_{i,\pi} \overset{(P2)}{=} 1,$$
$$\text{for each } i\in T,\ r\in R: \qquad \sum_{j\in T\setminus\{i\}} x_{\{i,j\},r} = \sum_{j\in T\setminus\{i\}}\sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi\in\Pi^{-i}} z_{i,\pi} \overset{(P2)}{=} 1.$$

We conclude the proof by constructing a feasible solution for the LP relaxation of the permutation formulation from a feasible solution x of the traditional formulation's LP relaxation.

Let x be such a solution and let $i\in T$. Consider the matrix $X^i\in\mathbb{R}^{(T\setminus\{i\})\times R}$ with entries $X^i_{j,r} = x_{\{i,j\},r}$. Due to all constraints of the traditional formulation's LP relaxation, X^i is a doubly stochastic matrix and is thus contained in the Birkhoff polytope, see [37]. Consequently, X^i can be written as a convex combination of all permutation matrices. That is, if $P^{i,\pi}$ is the permutation matrix associated with $\pi\in\Pi^{-i}$, there exist multipliers $\lambda^i_\pi\geq 0$, $\pi\in\Pi^{-i}$, such that $X^i = \sum_{\pi\in\Pi^{-i}}\lambda^i_\pi P^{i,\pi}$ and $\sum_{\pi\in\Pi^{-i}}\lambda^i_\pi = 1$. Based on these multipliers, we define a solution z of the permutation formulation via $z_{i,\pi} = \lambda^i_\pi$. To conclude the proof, we need to show that this solution z is feasible for the permutation formulation's LP relaxation and has the same objective value as x. Observe that z is non-negative since all λ's are non-negative. Constraints (P2) and (P3) are satisfied as

$$\text{for each } i\in T: \qquad \sum_{\pi\in\Pi^{-i}} z_{i,\pi} = \sum_{\pi\in\Pi^{-i}}\lambda^i_\pi = 1,$$
$$\text{for each } \{i,j\}\in\mathcal{M},\ r\in R: \qquad \sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} = \sum_{\pi\in\Pi^{-i}_{j,r}}\lambda^i_\pi = x_{\{i,j\},r} = \sum_{\pi\in\Pi^{-j}_{i,r}}\lambda^j_\pi = \sum_{\pi\in\Pi^{-j}} z_{j,\pi}$$

since $\sum_{\pi\in\Pi^{-i}}\lambda^i_\pi = 1$ and $x_{\{i,j\},r}$ is a convex combination of permutation matrices that assign team j (or i) to round r, respectively. Consequently, z is feasible for the permutation formulation's LP relaxation. Finally, both x and z have the same objective value because

$$\begin{aligned}\frac{1}{2}\sum_{i\in T}\sum_{\pi\in\Pi^{-i}} e_{i,\pi}z_{i,\pi} &= \frac{1}{2}\sum_{i\in T}\sum_{\pi\in\Pi^{-i}} e_{i,\pi}\lambda^i_\pi = \frac{1}{2}\sum_{i\in T}\sum_{\pi\in\Pi^{-i}}\sum_{r\in R} c_{\{i,\pi_r\},r}\lambda^i_\pi\\ &= \frac{1}{2}\sum_{i\in T}\sum_{j\in T\setminus\{i\}}\sum_{r\in R} c_{\{i,j\},r}\sum_{\substack{\pi\in\Pi^{-i}:\\ \pi_r=j}}\lambda^i_\pi P^{i,\pi}_{\pi_r,r} = \frac{1}{2}\sum_{i\in T}\sum_{j\in T\setminus\{i\}}\sum_{r\in R} c_{\{i,j\},r}x_{\{i,j\},r}\\ &= \sum_{\{i,j\}\in\mathcal{M}}\sum_{r\in R} c_{\{i,j\},r}x_{\{i,j\},r}.\end{aligned}$$

which proves that both formulations are relaxation-equivalent. □

Next, we turn our focus to the matching formulation and compare it with the traditional formulation (and thus, by the previous lemma, also with the permutation formulation).

Lemma 3.3. *For each $n\geq 6$, the matching formulation* (M) *is stronger than the traditional formulation* (T).

Proof. First, we show that we can transform any feasible solution of the matching formulation's LP relaxation to a feasible solution of the traditional formulation's LP relaxations. Afterwards, to show that the matching formulation is stronger than the traditional formulation, we show that, for any even $n\geq 6$, there exists an instance of SRR for which the LP relaxation of the matching formulation has a strictly larger value than the traditional formulation's LP relaxation.

Let y be a feasible solution of the matching formulation's LP relaxation. We construct a solution x for the traditional formulation by setting $x_{m,r} = \sum_{M\in\mathfrak{M}:\ m\in M} y_{m,r}$. Since y is non-negative, also x is non-negative. Moreover, Conditions (T2) and (T3) are satisfied as

$$\text{for each } m \in \mathcal{M}: \qquad \sum_{r\in R} x_{m,r} = \sum_{r\in R} \sum_{\substack{M\in\mathfrak{M}:\\ m\in M}} y_{M,r} \overset{\text{(M3)}}{=} 1,$$

$$\text{for each } i \in T,\ r \in R: \qquad \sum_{j\in T\setminus\{i\}} x_{\{i,j\},r} = \sum_{j\in T\setminus\{i\}} \sum_{\substack{M\in\mathfrak{M}:\\ \{i,j\}\in M}} y_{\{i,j\},r} = \sum_{M\in\mathfrak{M}} y_{M,r} \overset{\text{(M2)}}{=} 1.$$

Finally, both x and y have the same objective value as

$$\sum_{m\in\mathcal{M}} \sum_{r\in R} c_{m,r} x_{m,r} = \sum_{m\in\mathcal{M}} \sum_{r\in R} c_{m,r} \sum_{\substack{M\in\mathfrak{M}:\\ m\in M}} y_{M,r} = \sum_{M\in\mathfrak{M}} \sum_{r\in R} \sum_{m\in M} c_{m,r} y_{M,r} = \sum_{M\in\mathfrak{M}} \sum_{r\in R} d_{M,r} y_{M,r},$$

that is, the traditional formulation cannot be stronger than the matching formulation.

To prove that the matching formulation dominates the traditional formulation for $n \geq 6$ even, we distinguish three cases. In the first case, assume $n \geq 10$. Consider the pairs of teams given by

$$P = \big\{\{1,2\},\{2,3\},\{1,3\}\big\} \cup \big\{\{4,5\},\{5,6\},\{4,6\}\big\} \cup \big\{\{7,8\},\{8,9\},\dots,\{n-1,n\},\{7,n\}\big\}.$$

Interpreting P as the edges of an undirected graph, P defines three connected components consisting of two 3-cycles and an even cycle. We construct an instance of the SRR problem by specifying the cost function $c \in \mathbb{R}^{\mathcal{M}\times R}$ via

$$c_{m,r} = \begin{cases} 1, & \text{if } m \notin P \text{ and } r \in \{1,2\},\\ 0, & \text{otherwise.} \end{cases}$$

It is easy to verify that $x \in \mathbb{R}^{\mathcal{M}\times R}$ given by

$$x_{m,r} = \begin{cases} \frac{1}{2}, & \text{if } m \in P \text{ and } r \in \{1,2\},\\ 0, & \text{if } m \notin P \text{ and } r \in \{1,2\},\\ \frac{1}{n-3}, & \text{otherwise,} \end{cases}$$

is feasible for the LP relaxation of the traditional formulation and has objective value 0. Hence, x is optimal.

Solving the LP relaxation of the matching formulation for this instance, however, results in an objective value that is at least 2. Indeed, each perfect matching $M \in \mathfrak{M}$ contains at least one match $m \in M$ with $m \in \{\{i,j\} : (i,j) \in \{1,2,3\} \times \{4,5,\dots,n\}\}$. Since $c_{m,1} = c_{m,2} = 1$ for such a match, it follows that in both rounds 1 and 2, matchings are selected with total weight at least 1 due to (M3),leading to a solution with total cost at least 2.

In the second case, we consider $n = 6$. To prove the statement, we use the same construction as before, however, we do not require the even cycle anymore. That is, P defines two 3-cycles and the argumentation remains the same as before.

In the last case $n = 8$, we consider the set of pairs

$$P = \big\{\{1,2\},\{1,3\},\{2,3\}\big\} \cup \big\{\{4,5\},\{5,6\},\{6,7\},\{7,8\},\{4,8\}\big\}.$$

If we interpret P as edges of an undirected graph, the corresponding graph has two connected components being a 3-cycle and a 5-cycle, respectively. We choose the cost-coefficients $c \in \mathbb{R}^{\mathcal{M}\times R}$ to be

$$c_{m,r} = \begin{cases} 1, & \text{if } m \notin P \text{ and } r \in \{1,2\},\\ 0, & \text{otherwise.} \end{cases}$$

Simple calculations show that an optimal solution of the traditional formulation's LP relaxation is $x \in \mathbb{R}^{\mathcal{M}\times R}$ with

$$x_{m,r} = \begin{cases} \frac{1}{2}, & \text{if } m \in P \text{ and } r \in \{1,2\},\\ 0, & \text{if } m \notin E \text{ and } r \in \{1,2\},\\ \frac{1}{5}, & \text{otherwise,} \end{cases}$$

which has objective value 0, whereas the matching formulation's LP relaxation has value 2. $\square$

The previous two lemmata completely characterize the relative strength of the three different formulations except for $n = 4$. The status of this missing case is settled next.

Lemma 3.4. *For $n = 4$, the traditional formulation and the matching formulation are relaxation-equivalent.*

Proof. Observe that exactly the same arguments as in the proof of Lemma 3.3 can be used to show that the matching formulation is at least as strong as the traditional formulation if $n = 4$. Hence, it remains to show that every solution x of the traditional formulation's LP relaxation can be turned into a solution of the LP relaxation of the matching formulation if $n = 4$. Let x be such a solution. Let $M = \{\{i, j\}, \{k, l\}\} \in \mathfrak{M}$. Since x satisfies Equations (T3), summing the equations for i, j and subtracting the equations for k, l yields $2x_{\{i,j\},r} - 2x_{\{k,l\},r} = 0$. That is, the x-variables for the two matches of a matching within the same round have the same value. Consequently, the solution $y \in \mathbb{R}^{\mathfrak{M} \times R}$ given by $y_{M,r} = x_{\{i,j\},r}$ is well-defined and it is immediate to check that y is feasible for the matching formulation's LP relaxation and has the same objective value as x. $\square$

Summarizing the previous results of this section, we can provide a complete comparison of the strength of the traditional, matching, and permutation formulation.

Theorem 3.5. *For each $n \geq 6$, the traditional and permutation formulation are relaxation-equivalent, whereas the matching formulation is stronger than either of them. For $n = 4$, the traditional, matching, and permutation formulation for problem SRR are relaxation-equivalent.*

Besides verifying that all three models are equivalent for $n = 4$, we can also show that the matching formulation's integer hull is already completely characterized by (M2), (M3), as well as non-negativity inequalities for all variables.

Proposition 3.6. *For $n = 4$, Equations (M2) and (M3) as well as non-negativity inequalities define an integral polyhedron. That is, the matching formulation's LP relaxation coincides with its integer hull.*

Proof. To prove the proposition's statement, we show that the constraint matrix of (M) is totally unimodular. The result follows then by the Hoffman-Kruskal theorem [32] as all right-hand side values in (M) are integral.

For $n = 4$, the set of all matchings $\mathfrak{M}$ consists of exactly the three matchings

$$M_1 = \{\{1,2\},\{3,4\}\}, \qquad M_2 = \{\{1,3\},\{2,4\}\}, \qquad M_3 = \{\{1,4\},\{2,3\}\}.$$

The non-trivial constraints from Formulation (M) are Equations (M2) and (M3), which yield system

$$\begin{pmatrix} 1 & & & 1 & & & 1 & & \\ & 1 & & & 1 & & & 1 & \\ & & 1 & & & 1 & & & 1 \\ 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & & & \\ & & & & & & 1 & 1 & 1 \\ & & & & & & 1 & 1 & 1 \\ & & & 1 & 1 & 1 & & & \\ 1 & 1 & 1 & & & & & & \end{pmatrix} \begin{pmatrix} y_{M_1,1} \\ y_{M_1,2} \\ y_{M_1,3} \\ y_{M_2,1} \\ y_{M_2,2} \\ y_{M_2,3} \\ y_{M_3,1} \\ y_{M_3,2} \\ y_{M_3,3} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \qquad \begin{array}{l} ((\text{M2}), r = 1) \\ ((\text{M2}), r = 2) \\ ((\text{M2}), r = 3) \\ ((\text{M3}), m = \{1,2\}) \\ ((\text{M3}), m = \{1,3\}) \\ ((\text{M3}), m = \{1,4\}) \\ ((\text{M3}), m = \{2,3\}) \\ ((\text{M3}), m = \{2,4\}) \\ ((\text{M3}), m = \{3,4\}) \end{array}$$

Note that the last three equations are redundant and can be removed. The constraint matrix of the remaining equations is the node-edge incidence matrix of a bipartite graph and hence totally unimodular, which concludes the proof. $\square$

Thus, for $n = 4$, simply solving the LP-relaxation of the matching formulation by the simplex method, suffices to find an optimum integral solution.

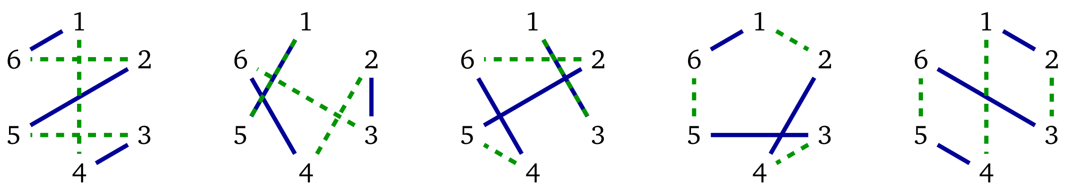
4 Strengthening the formulations

In this section, we continue our investigations of the structure of the formulations. In Section 4.1, we derive an exponentially sized class of valid inequalities for the matching formulation. Also, we show in Section 4.2 that adding the so-called odd-cut inequalities to the traditional formulation yields a formulation that is relaxation-equivalent to the matching formulation.

4.1 Strengthening the matching formulation

Observe that Theorem 3.5 does not rule out the possibility that, for $n \geq 6$, every vertex of the matching formulation is integral. That, however, is not the case already for $n = 6$ as we will show next. To this end, we first provide a fractional point $y^\star$ that is contained in the LP relaxation of the matching formulation for $n = 6$. Afterwards, we derive a class of valid inequalities for the integer hull matching formulation, and finally, we provide one such inequality that is violated by $y^\star$.

Example 4.1. Let $n = 6$. Then, the set of teams and rounds is given by $T = \{1, \dots, 6\}$ and rounds $R = \{1, \dots, 5\}$, respectively. In Figure 1, we depict a fractional solution of the matching formulation's LP relaxation. For each round $r \in R$, we provide two perfect matchings between the teams T, the blue and green (dashed) matching M, whose corresponding variables $y_{M,r}$ have value $\frac{1}{2}$ in the corresponding solution; all remaining variables have value 0. It is easy to verify that this fractional solution is indeed feasible for the LP relaxation of (M).



(a) round 1 **(b)** round 2 **(c)** round 3 **(d)** round 4 **(e)** round 5

Figure 1: A feasible point for the LP relaxation of Formulation (M).

To describe our class of valid inequalities, consider the following lemma.

Lemma 4.2. *Let $m_1, m_2 \in \mathcal{M}$ be disjoint and let $r' \in R$. Then,*

$$\sum_{r \in R \setminus \{r'\}} \sum_{\substack{M \in \mathfrak{M}: \\ m_1 \in M \text{ or } m_2 \in M}} y_{M,r} + \sum_{\substack{M \in \mathfrak{M}: \\ m_1 \notin M \text{ or } m_2 \notin M}} y_{M,r'} + \sum_{\substack{M \in \mathfrak{M}: \\ m_1, m_2 \in M}} 2y_{M,r'} \geq 2 \tag{4}$$

is a valid inequality for (M)*. In particular, it is a Chvátal-Gomory cut derived from the LP relaxation of* (M).

Proof. It is sufficient to prove that (4) is indeed a Chvátal-Gomory cut. To this end, we multiply Equation (M2) for round r' and Equations (M3) for matches m_1 and m_2 by $\frac{1}{2}$ and sum the resulting equations to obtain

$$\sum_{r \in R \setminus \{r'\}} \sum_{\substack{M \in \mathfrak{M}: \\ m_1 \in M \text{ or } m_2 \in M}} \frac{C_{M,r}}{2} y_{M,r} + \sum_{\substack{M \in \mathfrak{M}: \\ m_1 \notin M \text{ or } m_2 \notin M}} \frac{1 + C_{M,r}}{2} y_{M,r'} + \sum_{\substack{M \in \mathfrak{M}: \\ m_1, m_2 \in M}} \tfrac{3}{2} y_{M,r'} = \frac{3}{2},$$

where $C_{M,r} = |M \cap \{m_1, m_2\}|$. Since all y-variables are non-negative, we can turn this equation into a $\geq$-inequality by rounding up the left-hand side coefficients. Moreover, since in a feasible solution for (M) all variables attain integer values, we can increase the right-hand side from $\frac{3}{2}$ to 2, which yields the desired inequality. $\square$

Using this class of inequalities, we can show that the point $y^\star$ presented in the previous example is indeed not contained in the matching formulation's integer hull. Select $r' = 1$, $m_1 = \{1, 6\}$, and $m_2 = \{3, 5\}$, and let M be the blue and M' be the green matching of the first round as well as M'' the blue matching of round 4. Then, the corresponding inequality's left-hand side evaluates in $y^\star$ to $y^\star_{M'',4} + y^\star_{M,1} + y^\star_{M',1} = \frac{3}{2}$. Hence, $y^\star$ violates the corresponding inequality as $\frac{3}{2} \not\geq 2$.

Note that Inequality (4) is a so-called $\{0, \frac{1}{2}\}$-cut [7] as all multipliers used in the derivation are $\frac{1}{2}$ (and 0 for inequalities/equations that have not been used). By taking more equations in the generation of a valid inequality into account, we can generalize (4) to an exponentially large class of inequalities.

Proposition 4.3. *Let $A \subseteq \mathcal{M}$ be a set of pairwise disjoint matches and let $B \subseteq R$. If $|A| + |B|$ is odd, then*

$$\sum_{M \in \mathfrak{M}} \sum_{r \in B} \left\lceil \frac{1 + |M \cap A|}{2} \right\rceil y_{M,r} + \sum_{M \in \mathfrak{M}} \sum_{r \in R \setminus B} \left\lceil \frac{|M \cap A|}{2} \right\rceil y_{M,r} \geq \frac{1 + |A| + |B|}{2}, \tag{5}$$

is a valid inequality for (M). *In particular, it is a Chvátal-Gomory cut derived from the LP relaxation of* (M).

Proof. We follow the line of the proof of Lemma 4.2 and multiply each constraint of type (M2) with index in A and each constraint of type (M3) with index in B by $\frac{1}{2}$ and sum all resulting equations. This leads to

$$\sum_{j=1}^{n/2} \sum_{\substack{M \in \mathfrak{M}:\\ |M \cap A| = j}} \sum_{r \in B} \frac{1+j}{2} y_{M,r} + \sum_{j=1}^{n/2} \sum_{\substack{M \in \mathfrak{M}:\\ |M \cap A| = j}} \sum_{r \in R \setminus B} \frac{j}{2} y_{M,r} = \frac{|A| + |B|}{2}.$$

Since all y-variables are non-negative, we derive the inequality

$$\sum_{M \in \mathfrak{M}} \sum_{r \in B} \left\lceil \frac{1 + |M \cap A|}{2} \right\rceil y_{M,r} + \sum_{M \in \mathfrak{M}} \sum_{r \in R \setminus B} \left\lceil \frac{|M \cap A|}{2} \right\rceil y_{M,r} \geq \frac{|A| + |B|}{2},$$

and by integrality of the y-variables, we can round up the right-hand side, which leads to the desired inequality. ☐

While Inequalities (4) can trivially be separated in polynomial time, an efficient separation algorithm for (5) is not immediate. We leave the complexity status of separating (5) open for future research.

4.2 Strengthening the traditional formulation

Revisiting the proof of Lemma 3.3, it becomes clear that it is possible to assign, for a fixed round, each edge (match) of an odd cycle in K_n a weight of $\frac{1}{2}$. That is, the traditional formulation can assign an odd cycle of length k a weight of $\frac{k}{2}$. Such a solution, however, cannot be written as a convex combination of integer feasible solutions, because each such solution defines a perfect matching on the matches of a fixed round, i.e., the total weight of an odd cycle can be at most $\frac{k-1}{2}$. To strengthen the traditional formulation, one can thus add facet defining inequalities for the perfect matching polytope P_M to Model (T), which results in the additional inequalities

$$\sum_{i \in U} \sum_{j \in T \setminus U} x_{\{i,j\},r} \geq 1, \qquad U \subseteq T \text{ with } |U| \text{ odd}, r \in R, \tag{6}$$

which correspond to the odd-cut inequalities for the matching polytope and can be separated in polynomial time.

Lemma 4.4. *Let $n \geq 6$. The traditional formulation* (T) *extended by* (6) *is relaxation-equivalent to the matching formulation.*

Proof. We use the same proof strategy as for Lemma 3.3. Therefore, consider again the solution $x \in \mathbb{R}^{\mathcal{M} \times R}$ given by $x_{m,r} = \sum_{M \in \mathfrak{M} \colon m \in M} y_{M,r}$ for a solution y of the matching formulation's LP relaxation. Due to the proof of Lemma 3.3, it is sufficient to show that x satisfies (6) to prove that the matching formulation is at least as strong as the enhanced traditional formulation.

Let $U \subseteq T$ have odd cardinality. Since every $M \in \mathfrak{M}$ is a perfect matching, there is at least one team $i \in U$ that does not play against another team in U since U is odd. Hence, for each $M \in \mathfrak{M}$, there is a match $\{i,j\} \in M$ with $i \in U$ and $j \notin U$. Then,

$$\begin{aligned}\sum_{i\in U}\sum_{j\in T\setminus U} x_{\{i,j\},r} &= \sum_{i\in U}\sum_{j\in T\setminus U}\sum_{\substack{M\in\mathfrak{M}:\\ \{i,j\}\in M}} y_{M,r}\\ &= \sum_{M\in\mathfrak{M}}\sum_{i\in U}\sum_{\substack{j\in T\setminus U:\\ \{i,j\}\in M}} y_{M,r} \geq \sum_{M\in\mathfrak{M}} y_{M,r} \overset{\text{(M3)}}{\geq} 1.\end{aligned}$$

Consequently, the matching formulation is at least as strong as the enhanced traditional formulation.

To prove that the enhanced traditional formulation is not weaker than the matching formulation, we use a strategy similar to the one pursued in the proof of Lemma 3.2. Since the enhanced traditional formulation contains, per round r, all facet defining inequalities as well as equations for the perfect matching polytope P_M, each vector $X^r \in \mathbb{R}^{\mathcal{M}}$ given by $X^r_{\{i,j\}} = x_{\{i,j\},r}$ is contained in P_M. Hence, there exist non-negative multipliers $\lambda^r \in \mathbb{R}^{\mathfrak{M}}_+$ with $\sum_{M\in\mathfrak{M}} \lambda^r_M = 1$ such that $X^r = \sum_{M\in\mathfrak{M}} \lambda^r_M V_M$, where V_M is the vertex of P_M corresponding to the perfect matching M. We claim that $y \in \mathbb{R}^{\mathfrak{M}\times R}$ given by $y_{M,r} = \lambda^r_M$ is feasible for the LP relaxation of the matching formulation. Because $X^r = \sum_{M\in\mathfrak{M}} \lambda^r_M V_M$ implies $X^r_m = \sum_{M\in\mathfrak{M}\colon m\in M} \lambda^r_M$, both (M2) and (M3) are satisfied as

$$\begin{aligned}\sum_{M\in\mathfrak{M}} y_{M,r} &= \sum_{M\in\mathfrak{M}} \lambda^r_M = 1,\\ \sum_{\substack{M\in\mathfrak{M}:\\ m\in M}}\sum_{r\in R} y_{M,r} &= \sum_{\substack{M\in\mathfrak{M}:\\ m\in M}}\sum_{r\in R} \lambda^r_M = \sum_{r\in R} x_{m,r} \overset{\text{(T2)}}{=} 1.\end{aligned}$$

Moreover, y is non-negative as the λ's form a convex combination and both x and y have the same objective value since

$$\sum_{M\in\mathfrak{M}}\sum_{r\in R} d_{M,r} y_{M,r} = \sum_{M\in\mathfrak{M}}\sum_{r\in R}\sum_{m\in M} c_{m,r}\lambda^r_M = \sum_{m\in\mathcal{M}}\sum_{r\in R}\sum_{\substack{M\in\mathfrak{M}:\\ m\in M}} c_{m,r}\lambda^r_M = \sum_{m\in\mathcal{M}}\sum_{r\in R} c_{m,r} x_{m,r},$$

which concludes the proof. □

Remark 4.5. Since the traditional and permutation formulation are equivalent, one might wonder whether also the permutation formulation can be enhanced by odd-cut inequalities. Indeed, using the transformation $x_{\{i,j\},r} = \sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi}$ as in the proof of Lemma 3.2, one can show that the corresponding version of odd-cut inequalities is given by

$$\sum_{i\in U}\sum_{j\in T\setminus U}\sum_{\pi\in\Pi^{-i}_{j,r}} z_{i,\pi} \geq 1, \qquad U \subseteq T \text{ with } |U| \text{ odd}, r \in R,$$

and that the enhanced traditional and permutation formulation are equivalent.

5 An extension: k-round robin tournaments

In this section, we generalize the models for single round robin tournaments to k-round robin tournaments, where each pair of teams is required to meet exactly k times, for $k \geq 1$. As a consequence, the total number of matches that need to be scheduled becomes $\frac{1}{2}kn(n-1)$, and we set $R := \{1, 2, \dots, k(n-1)\}$.

Problem 2 (kRR)**.** Let $n \geq 4$ be an even integer and let $k \geq 1$ be integral. Given n teams with corresponding matches $\mathcal{M}$, a set of $\frac{1}{2}kn(n-1)$ rounds R ($k \geq 1$), as well as an integral cost $c_{m,r}$ for every $m \in \mathcal{M}$ and round $r \in R$, the *k-round robin (kRR) problem* is to find an assignment $\mathcal{A} \subseteq \mathcal{M} \times R$ of matches to rounds such that (i) every team plays a single match per round, (ii) each match is played in k, pairwise distinct, rounds, while total cost $\sum_{(m,r)\in\mathcal{A}} c_{m,r}$ is minimized.

Problem 1 (SRR) is a special case of kRR as it arises when $k = 1$. Another very prominent special case arises when $k = 2$, the so-called Double Round Robin tournament, denoted hereafter by DRR.

In principle, it is easy to generalize the models from Section 2 to account for meeting k times instead of once. Indeed, by replacing the right-hand side of constraints (T2), or the right-hand side of constraints (M3) by k, or by redefining Π^{-i} and $\Pi^{-i}_{j,r}$ to ordered lists for team i that features every opponent j exactly k times, the resulting formulations for kRR directly arise. In fact, we claim that it is not difficult to verify that the results concerning the polynomial solvability of the linear relaxations (Lemmata 2.1 and 2.2), as well as the strength of the relaxations (Theorem 3.5) hold for the kRR for each $k \geq 1$.

However, in practice, a number of additional properties become relevant when considering k-round robin tournaments: *phased* tournaments and tournaments where playing *home* or *away* matters. We now discuss these properties, and their consequences for the formulations, in more detail.

Phased (PH) The tournament is split into k parts such that each pair of teams meets once in each part. Here a *part* of the tournament refers to $n-1$ consecutive rounds, starting at round $\ell(n-1)+1$, for $\ell \in \{0,\dots,k-1\}$. Moreover, we use $R_\ell := \{\ell(n-1)+1,\dots,(\ell+1)(n-1)\}$ to denote the rounds in part $\ell \in \{0,\dots,k-1\}$, and $R := \bigcup_{\ell=0}^{k-1} R_\ell$.

Without the presence of any additional constraints, a phased tournament can be trivially decomposed in multiple single-round robin tournaments: one for each set of rounds R_ℓ.

Home-away (HA) Each team has a home venue, implying that to specify a schedule it is no longer sufficient to specify the matches in each round; instead, one also has to specify, for each match, which teams plays home, and which team plays away. We denote this by redefining a match between teams $i, j \in T$ ($i \neq j$) where i is the home-playing team, by an ordered pair (i, j) (in contrast to an unordered pair $\{i, j\}$).

Let k be a positive integer. For n teams and a k-round robin setting, denote $T := \{1,\dots,n\}$ and $R := \{1,\dots,k(n-1)\}$. Let $\mathcal{M} := \{(i,j) : i,j \in T, i \neq j\}$ be the set of ordered matches. The assignment of match $(i,j) \in \mathcal{M}$ to round $r \in R$ comes at a cost $c_{(i,j),r}$, and in contrast to the SRR case, $c_{(i,j),r}$ and $c_{(j,i),r}$ can be different. We proceed by describing the phased k-round robin problem with home-away patterns (kRR-PH-HA):

Problem 3 (kRR-PH-HA)**.** Given an even number $n \geq 4$ of teams with corresponding matches $\mathcal{M}$, a set of $\frac{1}{2}kn(n-1)$ rounds R ($k \geq 1$), as well as an integral cost $c_{m,r}$ for every $m \in \mathcal{M}$ and round $r \in R$, the kRR-PH-HA problem is to find an assignment $\mathcal{A} \subseteq \mathcal{M} \times R$ of matches to rounds such that (i) every team plays a single match per round, (ii) each pair of teams meets once in part $R_\ell, \ell \in \{1,\dots,k\}$, and (iii) each ordered match is played $\lfloor \frac{k}{2} \rfloor$ or $\lceil \frac{k}{2} \rceil$ times so that each pair of teams meets in total k times, while total cost $\sum_{(m,r)\in\mathcal{A}} c_{m,r}$ is minimized.

We will show how the three formulations of Sections 2.1–2.3 can be adapted to deal with these properties.

Extending the traditional formulation for k-RR tournaments For reasons of convenience, we assume k is even; this implies that for each pair of distinct teams $i, j \in T$ match (i, j) and match (j,i) each need to occur $\frac{k}{2}$ times in any feasible schedule.

$$\min \sum_{(i,j)\in\mathcal{M}} \sum_{r\in R} c_{(i,j),r} x_{(i,j),r} \tag{kT1}$$

$$\sum_{r\in R} x_{(i,j),r} = \frac{k}{2}, \qquad (i,j)\in\mathcal{M}, \tag{kT2}$$

$$\sum_{r\in R_\ell} (x_{(i,j),r} + x_{(j,i),r}) = 1, \qquad i,j\in T,\ i\neq j,\ \ell\in\{0,\dots,k-1\}, \tag{kT3}$$

$$\sum_{j\in T\setminus\{i\}} (x_{(i,j),r} + x_{(j,i),r}) = 1, \qquad i\in T,\ r\in R, \tag{kT4}$$

$$x_{(i,j),r}\in\{0,1\}, \qquad (i,j)\in\mathcal{M},\ r\in R. \tag{kT5}$$

Constraints (kT2) ensure that each match is played $\frac{k}{2}$ times, Constraints (kT3) express that each pair of teams has to meet once in each part (the "phased" property), and Constraints (kT4) prescribe that each team plays a single match in each round.

Extending the matching formulation for kRR tournaments We now assume that $K_n = (T, A)$ is a *directed* multi-graph, where each arc (i,j) with $i,j\in T$, $i\neq j$ is present $\frac{k}{2}$ times; a (directed) matching M is now defined as a set of $\frac{n}{2}$ arcs incident to each node once, and $\mathfrak{M}$ now stands for the set of all (directed) matchings.

$$\min \sum_{M\in\mathfrak{M}} \sum_{r\in R} d_{M,r} y_{M,r} \tag{kM1}$$

$$\sum_{M\in\mathfrak{M}} y_{M,r} = 1, \qquad r\in R, \tag{kM2}$$

$$\sum_{r\in R} \sum_{\substack{M\in\mathfrak{M}:\\ (i,j)\in M}} y_{M,r} = \frac{k}{2}, \qquad (i,j)\in\mathfrak{M}, \tag{kM3}$$

$$\sum_{r\in R_\ell} \sum_{\substack{M\in\mathfrak{M}:\\ (i,j)\in M \text{ or } (j,i)\in M}} y_{M,r} = 1, \qquad i,j\in T,\ i\neq j,\ \ell\in\{0,\dots,k-1\}, \tag{kM4}$$

$$y_{M,r}\in\{0,1\}, \qquad M\in\mathfrak{M},\ r\in R. \tag{kM5}$$

Constraints (kM2) ensure that a (directed) perfect matching is selected in each round, Constraints (kM3) say that each match occurs $\frac{k}{2}$ times, and Constraints (kM4) model the fact that each pair of teams has to meet once in each part.

Extending the permutation formulation for kRR tournaments We have to redefine Π^{-i}: each entry needs to specify home or away, and of course, the fact that each team meets all other teams in rounds $(\ell-1)(n-1)+1,\dots,\ell(n-1)$, for each $\ell\in\{1,\dots,k\}$ needs to be taken into account. Other than that Formulation (P) remains unchanged.

Without giving formal proofs, we claim that the linear relaxations of the extension of the matching formulation, as well as the linear relaxation of the extension of the permutation formulation can be solved in polynomial time. We also claim that the extension of the matching formulation is stronger than the other two formulations—the adaptations to the proofs of Lemmata 2.1 and 2.2 and Theorem 3.5 are straightforward.

6 Computational results

In this section, we report the outcomes of our computational experiments. Section 6.1 describes the test set that we have used in our experiments. Afterwards, we investigate the quality of the LP relaxations of the different models and compare their corresponding values in Section 6.2.

Finally, we discuss our experience with solving instances of the SRR problem using the matching formulation, i.e., not only solving the LP relaxation but also the corresponding integer program. To this end, we have implemented a branch-and-price algorithm whose details are described in Section 6.3.

6.1 Test set

We have generated 1000 instances[1] of the SRR problem to evaluate the quality of the LP relaxations of the different models. Our test set comprises of instances of different sizes and thus different levels of difficulty, which are parameterized by a tuple (n, ρ) and have cost coefficients attaining values 0 or 1. Parameter n encodes the number of teams and has range $n \in \{6, 12, 18, 24\}$; parameter ρ controls the number of 1-entries in the objective. More precisely, we pick a set of match-round pairs $\mathcal{M} \times R$ of size $\lfloor \rho \cdot |\mathcal{M} \times R| \rfloor$ uniformly at random, denoted by $S \subseteq \mathcal{M} \times R$, where $\rho \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. The generated instance consists of n teams and has cost coefficients $c_{m,r} = 1$ if $(m, r) \in S$ and $c_{m,r} = 0$ otherwise. For each combination of $n \in \{6, 12, 18, 24\}$ and $\rho \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ we have generated 50 instances.

6.2 A computational comparison of the linear relaxations

In this section, we provide a computational comparison between the LP relaxation values of the traditional formulation (T) (and thus by Lemma 3.2 also of the permutation formulation) and LP relaxation values of the matching formulation (M), and compare these to the actual optimal (or best found) integral solutions. Before we discuss our numerical results, we provide details about our implementation as well as on how we find optimal integral solutions first.

Implementation details To find the LP relaxation values, we implemented both formulations in Python 3 using the `PySCIPOpt` 4.1.0 package [24] for `SCIP` 8.0.0 [3], with `CPLEX` 20.1.0.0 as LP solver. The traditional formulation is implemented as a compact model. For the matching formulation, we use a column generation procedure that receives a subset of all variables, solves the corresponding LP relaxation restricted to these variables, and adds further variables until it can prove that an optimal LP solution has been found. To identify whether new variables need to be added, we solve the so-called pricing problem, which corresponds to separating a corresponding solution of the dual problem. The separation problem can be solved by finding a maximum weight perfect matching as detailed in the proof of Lemma 2.1. We start with the empty set of variables, which means that the primal problem is initially infeasible. Analogously to the proof of Lemma 2.1, we resolve infeasibility by adding variables to the problem that are associated with a dual constraint that violate a dual unbounded ray of this infeasible problem. The column generation procedure has been embedded in a so-called `pricer` plug-in of `SCIP`, which adds newly generated variables to the matching formulation. The maximal weight perfect matchings are computed using `NetworkX` 2.5.1, which provides an implementation of Edmonds' blossom algorithm.

Finding optimal integer solutions To obtain the optimal integer solution value of as many instances as possible, we have used two different solvers to solve the integer program of Model (T). On the one hand, we have used `SCIP` as described in the above setup. On the other hand, we have modeled (T) using `Gurobi` 9.1.2 via its Python 3 interface. For each instance and solver, we have imposed a time limit of 48 h to find an optimal integer solution. Using `SCIP`, we managed to solve 852 of the 1000 instances to optimality. With `Gurobi`, we were able to solve 866 of the 1000 instances to optimality. There were 45 instances where `SCIP` found a better primal objective value, and 79 instances where `Gurobi` found a better primal objective value. All experiments have been run on a compute cluster with identical machines, using one (resp. two) thread(s) on Xeon Platinum 8260 processors, with 10.7 GB (resp. 21.4 GB) memory, respectively for `SCIP` and `Gurobi`.

[1]The instances as well as the implementation of our algorithms are publicly available at `https://github.com/JasperNL/round-robin`; all experiments have been conducted using the code with githash `1657b4d7`.

Table 1: Comparison of the LP relaxation values of the traditional and matching formulation.

		all instances					restricted to instances with $v^{\text{LP}}_{\text{tra}} < v^{\text{IP}}$							
		average value			solved			average value			solved		gap closed	
n	ρ	$v^{\text{LP}}_{\text{tra}}$	$v^{\text{LP}}_{\text{mat}}$	v^{IP}	O	T	#	$v^{\text{LP}}_{\text{tra}}$	$v^{\text{LP}}_{\text{mat}}$	v^{IP}	O	T	average	maximal
6	0.5	2.227	2.297	2.380	50	0	14	2.452	2.702	3.000	14	0	43.45%	100.00%
6	0.6	3.802	3.865	3.920	50	0	12	3.924	4.188	4.417	12	0	52.08%	100.00%
6	0.7	5.430	5.510	5.540	50	0	9	5.611	6.056	6.222	9	0	66.67%	100.00%
6	0.8	7.620	7.635	7.660	50	0	4	7.250	7.438	7.750	4	0	37.50%	100.00%
6	0.9	10.003	10.040	10.060	50	0	6	10.361	10.667	10.833	6	0	66.67%	100.00%
12	0.5	0.080	0.080	0.080	50	0	0	–	–	–	0	0	–	–
12	0.6	2.018	2.213	3.480	50	0	49	2.019	2.217	3.510	49	0	15.29%	100.00%
12	0.7	8.022	8.342	9.500	50	0	50	8.022	8.342	9.500	50	0	22.27%	70.77%
12	0.8	17.184	17.474	18.340	50	0	50	17.184	17.474	18.340	50	0	29.89%	100.00%
12	0.9	31.459	31.654	31.840	50	0	31	31.096	31.410	31.710	31	0	56.87%	100.00%
18	0.5	0.000	0.000	0.000	50	0	0	–	–	–	0	0	–	–
18	0.6	0.060	0.060	0.060	50	0	0	–	–	–	0	0	–	–
18	0.7	2.045	2.292	5.600	50	0	50	2.045	2.292	5.600	50	0	6.68%	15.66%
18	0.8	19.831	20.330	23.900	50	0	50	19.831	20.330	23.900	50	0	12.37%	28.19%
18	0.9	52.700	53.066	54.500	50	0	49	52.673	53.047	54.510	49	0	21.55%	100.00%
24	0.5	0.000	0.000	0.000	50	0	0	–	–	–	0	0	–	–
24	0.6	0.000	0.000	0.000	50	0	0	–	–	–	0	0	–	–
24	0.7	0.200	0.200	4.340	0	50	49	0.163	0.163	4.408	0	49	0.00%	0.00%
24	0.8	12.352	12.893	24.180	0	50	50	12.352	12.893	24.180	0	50	4.57%	7.91%
24	0.9	69.327	69.922	74.860	29	21	50	69.327	69.922	74.860	29	21	10.69%	21.68%

Numerical results Table 1 shows the aggregated computational results of our experiments. For each number of teams n and ratio ρ, we provide the average of the objective values of the relaxation of the traditional formulation (column "$v^{\text{LP}}_{\text{tra}}$"), the average of the objective values of the relaxation of the matching formulation (column "$v^{\text{LP}}_{\text{mat}}$"), and the average optimum value (column "v^{IP}"). Notice that for $n = 24$ we have not been able to solve all instances to optimality; in this case, we use the value of the best known solution instead of the (unknown) optimum for that instance in the v^{IP} column. Recall that each value is an average over 50 instances. The number of optimally solved instances (resp. instances not terminating within the time limit) are shown in column "O" (resp. "T").

To be able to assess the strength of the matching formulation compared to the traditional formulation, we focus, in the right side of the table, on those instances for which $v^{\text{LP}}_{\text{tra}} < v^{\text{IP}}$; their number (out of 50) is given in the column labeled "#". From this column, we see that the fraction ρ that leads to instances with a gap between $v^{\text{LP}}_{\text{tra}}$ and v^{IP} slowly increases with n. Indeed, for $n = 6$, most instances do not have a gap, for $n = 12$, almost all instances with $\rho \in \{0.6, 0.7, 0.8\}$ have a gap, and for $n = 18$, almost all instances with $\rho \in \{0.7, 0.8, 0.9\}$ have a gap.

We use the notion of the *relative gap* that is closed by the matching formulation relative to the traditional formulation, given by

$$\text{rgap}(I) := \frac{v^{\text{LP}}_{\text{mat}}(I) - v^{\text{LP}}_{\text{tra}}(I)}{v^{\text{IP}}(I) - v^{\text{LP}}_{\text{tra}}(I)} \text{ for an instance } I \text{ of SRR with } v^{\text{IP}}(I) - v^{\text{LP}}_{\text{tra}}(I) > 0.$$

A value of zero for rgap(I) implies that the relaxation values of the traditional formulation and the matching formulation are equal, while a value of one (i.e., 100%) implies that the relaxation of the matching formulation is equal to the true objective of the optimal integral solution. The column "average" gives the average rgap, whereas column "maximal" shows the maximum relative closed gap for an instance of this sub test set.

For $n = 6$, there are few instances with a gap. However, for those instances for which there is a gap, it is clear that a sizable part of that gap is closed by the relaxation of the matching formulation. For larger values of n, many instances have a gap. We observe that a significant percentage of the gap is closed by the relaxation of the matching formulation. If n is getting larger, however, both the value of the average gap closed as well as the value of the maximal gap closed decrease. We conclude that for small values of n, and thus for many realistic applications, the matching formulation provides a much better relaxation value than the traditional formulation.

6.3 A branch-and-price algorithm

Since the matching formulation can dominate the traditional formulation, a natural question is whether the stronger formulation also allows to solve the SRR problem faster than the traditional formulation. For this reason, we have implemented a branch-and-price algorithm (in the computational setup as described above) to compute optimal integral solutions of the matching formulation. That is, we use a branch-and-bound algorithm to solve the matching formulation, where each LP relaxation is solved using a column generation procedure.

Implementation details In classical branch-and-bound algorithms, the most common way to implement the branching scheme is to select a variable x_i whose value $x_i^\star$ in the current LP solution is non-integral and to generate two subproblems by additionally enforcing either $x_i \leq \lfloor x_i^\star \rfloor$ or $x_i \geq \lceil x^\star \rceil$. In principle, this strategy is also feasible for the matching formulation, where the subproblems correspond to forbidding a schedule $M \in \mathfrak{M}$ for a round $r \in R$ or fixing the schedule in round r to be M. This branching scheme, however, leads to a very unbalanced branch-and-bound tree as the former subproblem only rules out a very specific schedule, while the latter one fixes the matches of an entire round. Another difficulty of the classical scheme is that it might affect the structure of the pricing problem in the newly generated subproblems. Ideally, the pricing problem should not change such that the same algorithm can be used for adding new variables to the problem. We will address both issues next.

To obtain a more balanced branch-and-bound tree, we have implemented a custom branching rule following the Ryan-Foster branching scheme [31]: Our scheme selects a match $\{i, j\} \in \mathcal{M}$ at a round $r \in R$ and creates two children. In the left child, we forbid that $\{i, j\}$ is played in round r, and in the right child, we enforce that $\{i, j\}$ is played in round r. Note that for all matchings $M \in \mathfrak{M}$ this branching decision fixes all variables $y_{M,r}$ to zero if $\{i, j\} \in M$ for the left child, and $\{i, j\} \notin M$ for the right child.

Using this branching strategy, the structure of the pricing problem at each subproblem remains a matching problem. At the root node of the branch-and-bound tree, we need to solve a maximum weight perfect matching problem in a weighted version of K_n as described above. At other nodes of the branch-and-bound tree, we have added branching decisions that enforce that two teams i and j either do meet or do not meet in a round $r \in R$. These decisions can easily be incorporated by deleting edges from K_n. When generating variables for round r, we remove edge $\{i, j\}$ from K_n if i and j shall not meet in this round; if the match $\{i, j\}$ shall take place, then we remove all edges incident with i and j except for $\{i, j\}$. Consequently, our branching strategy allows to solve the LP relaxations of all subproblems in polynomial time.

Since our Python implementation of the traditional and matching formulation took too much time to be used in a branching scheme, we decided to implement our branch-and-price algorithm as a plug-in using the C-API of `SCIP`. The pricer plug-in is analogous, and maximal weight perfect matchings are now computed using the `LEMON` 1.3.1 graph library. To ensure that the branching decisions are taken into account, we also implemented a constraint that fixes $y_{M,r}$ to zero if the matching M violates the branching decisions for round r, and added a plug-in that implements the branching decisions.

The branching rule sketched above admits some degrees of freedom in selecting the match $\{i, j\}$ and round r. In our implementation, we decided to mimic two well-known branching rules: most infeasible branching and strong branching on a selection of variables, see Achterberg et al. [1] for an overview on branching rules. Most infeasible branching branches on a binary variable with fractional value in an LP solution that is closest to 0.5, and strong branching branches on the variable that yields the largest dual bound improvement based on some metric. Since strong branching requires significant computational effort, it is common to make a limited branching candidate selection and apply strong branching on those.

The pseudocode for our branching rule is given in Algorithm 1. We start by computing the fractional match-on-round assignment values induced by the y-variables. Then, we make a selection of potentially good branching candidates $(m, r) \in \mathcal{M} \times R$, that is based on the $\text{score}_{m,r}$ metric shown in Line 8: this score prioritizes match-on-round assignments for which (i) the assignment value is close to 0.5, (ii) the cost coefficients are large, and (iii) the assignment values are relatively high. Using this score, we hope to resolve fractionality soon (by (i)). By (ii), we want to enforce

Algorithm 1: Determining the branching candidate for an LP node.

```
input  : An LP solution $y^\star_{m,r}$ in a branch-and-bound tree node at depth $d$ with objective obj.
output: The branching decision (match $m \in \mathcal{M}$ on round $r \in R$), or detected integrality.
1  // fractional assignment of match $m$ to round $r$
2  compute assign$_{m,r} \leftarrow \sum_{M \in \mathfrak{M}\colon m \in M} y^\star_{M,r}$ for $m \in \mathcal{M}$ and $r \in R$;
3  if assign$_{m,r}$ is $0$ or $1$ for all $m \in \mathcal{M}$ and $r \in R$ then
4      return integral solution found;
5  // fractional part of assign$_{m,r}$
6  compute frac$_{m,r} \leftarrow \min\{$assign$_{m,r}, 1.0 -$ assign$_{m,r}\}$ for $m \in \mathcal{M}$ and $r \in R$;
7  // score for every match-round pair
8  compute score$_{m,r} \leftarrow$ frac$_{m,r} \cdot (1.0 + |c_{m,r}|) \cdot ($assign$_{m,r})^2$ for $m \in \mathcal{M}$ and $r \in R$ ;
9  // strong branching candidate selection
10 number_of_candidates $\leftarrow \max\{1, \lfloor 0.1 \cdot |\mathcal{M} \times R| \cdot 0.65^d \rfloor\}$;
11 if number_of_candidates $> 1$ then
12     pick number_of_candidates candidates $(m, r) \in \mathcal{M} \times R$ with highest score$_{m,r}$ as strong
         branching candidates;
13     foreach strong branching candidate $(m, r)$ do
14         if score$_{m,r} = 0.0$ then
15             // then assign$_{m,r}$ is 0 or 1, skip this candidate
16             continue (i.e., skip this candidate);
17         apply strong branching on $(m, r)$, with objectives obj$_{\text{forbid}}$, obj$_{\text{enforce}}$ in the two
             children;
18         compute score$^\star_{m,r} \leftarrow ($obj$_{\text{forbid}} -$ obj $+ 1.0) \cdot ($obj$_{\text{enforce}} -$ obj $+ 1.0)$;
19     return branch on strong branching candidate $(m, r)$ with maximal score$^\star_{m,r}$;
20 else
21     // do not apply strong branching deep in the branch-and-bound tree
22     return branch on $(m, r) \in \mathcal{M} \times R$ with maximal score$_{m,r}$;
```

Table 2: Computational results for the branch-and-price algorithm for Model (M).

			Solved		Solving time (s)		
n	ρ	#	O	T	min	mean	max
6	0.5	50	50	0	0.00	0.00	0.01
6	0.6	50	50	0	0.00	0.00	0.01
6	0.7	50	50	0	0.00	0.00	0.01
6	0.8	50	50	0	0.00	0.00	0.01
6	0.9	50	50	0	0.00	0.00	0.01
12	0.5	50	50	0	1.25	2.38	5.73
12	0.6	50	50	0	0.12	4.09	8.28
12	0.7	50	50	0	0.21	3.33	7.38
12	0.8	50	50	0	0.14	2.05	7.63
12	0.9	50	50	0	0.11	0.54	2.21
18	0.5	50	50	0	54.61	106.09	210.77
18	0.6	50	48	2	103.41	866.21	7200.00
18	0.7	50	3	47	930.53	6854.47	7200.09
18	0.8	50	22	28	312.65	4084.21	7200.06
18	0.9	50	50	0	6.74	332.98	3990.61

a significant change of the objective value in the child that forbids match m, whereas the child enforcing that m is played selects a match that is most likely played due to (iii). Experiments show that full strong branching leads to a smaller number of nodes to solve the problem, but this turns out to be very costly computationally. Therefore, we only apply strong branching for branch-and-bound tree nodes close to the root, and only evaluate a subset of branching candidates that have the highest score$_{m,r}$-metric. This is our candidate pre-selection. The higher the depth of the considered branch-and-bound tree node, the smaller the number of candidates considered. Of those branching candidates, we pick the candidate that maximizes score$^{\star}_{m,r}$ as defined in Line 18. The goal of this score is to choose the candidate where the objective values of the hypothetical children are different from the current node's objective. By considering the product of this difference, we prioritize if the objective of both hypothetical children have some difference with the current node objective. If the number of candidates is only one, no strong branching is applied and that candidate match-on-round assignment is chosen for branching.

All experiments have been run on a Linux cluster with Intel Xeon E5 3.5 GHz quad core processors and 32 GB memory. The code was executed using a single thread and the time limit for all computations was 2 h per instance.

Numerical results Table 2 summarizes our results for our instances for $n \in \{6, 12, 18\}$. We distinguish the instances by their parameters n and ρ, and we report on the number of instances that could be solved (resp. could not be solved) within the time limit in column "O" (resp. "T"). Moreover, we report on the the minimum, mean, and maximum running time per parameterization. The mean of all running times t_i is reported in shifted geometric mean $\prod_{i=1}^{50}(t_i + s)^{\frac{1}{50}} - s$ using a shift of 10 s to reduce the impact of instances with very small running times.

We observe that instances with 6 and 12 teams can be solved very efficiently within fractions of seconds in the former and within seconds in the latter case. Instances with 18 teams are more challenging, in particular, if the ratio $\rho \in \{0.7, 0.8\}$. In this case, only 3 and 22 instances could be solved, respectively, but note that not all instances are equally difficult. For instance, for $n = 18$ and $\rho = 0.8$, there exists an instance that can be solved within roughly five minutes, whereas the mean running time is more than an hour. To fully benefit from the strong LP relaxation of the matching formulation, it might be the case that additional algorithmic enhancements can further improve the performance of the branch-and-price algorithm.

7 Conclusion

The use of integer programming for finding schedules of round robin tournaments is widespread. We have introduced and analyzed two new formulations for this problem, one of which (the matching formulation) is stronger than the other formulations. We have proposed a class of valid inequalities for the matching formulation, which may be of use when developing cutting-plane based techniques for this problem. By randomly generating instances, we studied the strength of the formulations, and we implemented a branch-and-price algorithm based on the matching formulation to see its efficiency. Although this algorithm is able to solve small-scale instances rather efficiently, solving large instances of the SRR efficiently remains a challenge.

Possible directions of future research are thus to further strengthen our integer programming formulations/techniques. On the one hand, once can investigate additional cutting planes to strengthen both the traditional and matching formulation. For the matching formulation, cutting planes will in particular affect the pricing problem and thus might change its structure. Thus, the trade-off between the strength of cutting planes and the difficulty of solving the pricing problem that needs to be investigated. On the other hand, one can enhance our branch-and-price algorithm in several directions, e.g., the development of more sophisticated branching rules or heuristics for producing good schedules.

Acknowledgment The fourth author's research is supported by the Dutch Research Council (NWO) through Gravitation grant NETWORKS-024.002.003.

References

[1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[2] F. Alarcón, G. D. M. Guajardo, J. Miranda, H. Munoz, L. Ramirez, M. Ramirez, D. Saure, M. Siebert, S. Souyris, A. Weintraub, R. Wolf-Yadlin, and G. Zamoranoa. Operations research transforms the scheduling of chilean soccer leagues and south american world cup qualifiers. *Interfaces*, 47:52–69, 2017.

[3] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021.

[4] E. L. Bouzarth, B. C. Grannan, J. M. Harris, and K. R. Hutson. Scheduling the valley baseball league. *INFORMS Journal on Applied Analytics*, 2021.

[5] D. Briskorn and A. Drexl. IP models for round robin tournaments. *Computers & Operations Research*, 36(3):837–852, 2009.

[6] D. Briskorn, A. Drexl, and F. C. Spieksma. Round robin tournaments and three index assignments. *4OR*, 8(4):365–374, 2010.

[7] A. Caprara and M. Fischetti. {0, 1/2}-Chvátal-Gomory cuts. *Mathematical Programming*, 74(3):221–235, 1996.

[8] G. Cocchi, A. Galligari, F. Nicolino, V. Piccialli, F. Schoen, and M. Sciandrone. Scheduling the Italian National Volleyball Tournament. *Interfaces*, 48:271–284, 2018.

[9] F. D. Croce and D. Oliveri. Scheduling the Italian Football League: an ILP-based approach. *Computers and Operations Research*, 33:1963–1974, 2006.

[10] G. Durán, M. Guajardo, F. Gutiérrez, J. Marenco, D. Sauré, and G. Zamorano. Scheduling the main professional football league of Argentina. *INFORMS Journal on Applied Analytics*, 51(5):361–372, 2021.

[11] G. Duran, M. Guajardo, J. Miranda, D. Saure, S. Souyris, A. Weintraub, and R. Wolf. Scheduling the Chilean soccer league by integer programming. *Interfaces*, 37:539–552, 2007.

[12] G. Durán, M. Guajardo, and D. Saure. Scheduling the South American Qualifiers to the 2018 FIFA World Cup by integer programming. *European Journal of Operational Research*, 262:1109–1115, 2017.

[13] K. K. Easton. *Using integer programming and constraint programming to solve sports scheduling problems*. PhD thesis, Georgia Institute of Technology, 2003.

[14] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

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

[16] C. Fleurent and J. Ferland. Allocating games for the NHL using integer programming. *Operations Research*, 41:649–654, 1993.

[17] D. Goossens and F. Spieksma. Scheduling the Belgian soccer league. *Interfaces*, 39:109–118, 2009.

[18] D. Goossens and F. Spieksma. Soccer schedules in Europe: an overview. *Journal of Scheduling*, 15:641–651, 2012.

[19] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

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

[21] T. Kim. Optimal approach to game scheduling of multiple round-robin tournament: Korea professional baseball league in focus. *Computers and Industrial Engineering*, 136:95–105, 2019.

[22] S. Knust. Classification of literature on sports scheduling. http://www.inf.uos.de/knust/sportssched/sportlit_class/. Accessed: March 2022.

[23] K. Kostuk and K. Willoughby. A decision support system for scheduling the Canadian football league. *Interfaces*, 42:286–295, 2012.

[24] S. Maher, M. Miltenberger, J. P. Pedroso, D. Rehfeldt, R. Schwarz, and F. Serrano. PySCIPOpt: Mathematical programming in python with the SCIP optimization suite. In *Mathematical Software – ICMS 2016*, pages 301–307. Springer International Publishing, 2016.

[25] G. Nemhauser and M. Trick. Scheduling a major college basketball conference. *Operations Research*, 46:1–8, 1998.

[26] M. Raknes and K. Pettersen. Optimizing sports scheduling: Mathematical and constraint programming to minimize traveled distance with benchmark from the Norwegian professional volleyball league. Master's thesis, Norwegian Business School, 2018.

[27] R. Rasmussen. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research*, 185:795–810, 2008.

[28] R. V. Rasmussen and M. A. Trick. Round robin scheduling–a survey. *European Journal of Operational Research*, 188(3):617–636, 2008.

[29] D. Recalde, R. Torres, and P. Vaca. Scheduling the professional Ecuadorian football league by integer programming. *Computers and Operations Research*, 40:2478–2484, 2013.

[30] C. Ribeiro and S. Urrutia. Scheduling the Brazilian soccer tournament: solution approach and practice. *Interfaces*, 42:260–272, 2012.

[31] D. Ryan and B. Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer scheduling of public transport: Urban passenger vehicle and crew scheduling*, pages 269–280. North-Holland, 1981.

[32] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1987.

[33] M. A. Trick. Integer and constraint programming approaches for round-robin tournament scheduling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 63–77. Springer, 2002.

[34] D. Van Bulck and D. Goossens. On the complexity of pattern feasibility problems in time-relaxed sports timetabling. *Operations Research Letters*, 48(4):452–459, 2020.

[35] D. van Bulck, D. Goossens, J. Beliën, and M. Davari. The fifth international timetabling competition (ITC 2021): Sports timetabling. In *Proceedings of MathSport International 2021 Conference, MathSport*, pages 117–122, 2021.

[36] S. Westphal. Scheduling the German basketball league. *Interfaces*, 44:498–508, 2014.

[37] G. M. Ziegler. *Lectures on Polytopes*. Springer, New York, graduate texts in mathematics 152 edition, 1995.