European Journal of Operational Research 259 (2017) 1180–1190

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Scheduling double round-robin tournaments with divisional play using constraint programming

Mats Carlsson[a,*], Mikael Johansson[b], Jeffrey Larson[c]

[a] SICS, P.O. Box 1263, Kista SE-164 29, Sweden
[b] Automatic Control Lab, KTH, Osquldas väg 10, Stockholm SE-100 44, Sweden
[c] MCS Division, Argonne National Laboratory, Lemont, IL 60439, USA

ARTICLE INFO

Article history:
Received 29 January 2016
Accepted 16 November 2016
Available online 1 December 2016

Keywords:
OR in sports
Scheduling
Constraint programming

ABSTRACT

We study a tournament format that extends a traditional double round-robin format with divisional single round-robin tournaments. Elitserien, the top Swedish handball league, uses such a format for its league schedule. We present a constraint programming model that characterizes the general double round-robin plus divisional single round-robin format. This integrated model allows scheduling to be performed in a single step, as opposed to common multistep approaches that decompose scheduling into smaller problems and possibly miss optimal solutions. In addition to general constraints, we introduce Elitserien-specific requirements for its tournament. These general and league-specific constraints allow us to identify implicit and symmetry-breaking properties that reduce the time to solution from hours to seconds. A scalability study of the number of teams shows that our approach is reasonably fast for even larger league sizes. The experimental evaluation of the integrated approach takes considerably less computational effort to schedule Elitserien than does the previous decomposed approach.

1. Introduction

Double round-robin tournaments (DRRTs), competitions where every team plays every other team once at home and once away, are one of the most common formats for a broad range of sporting events. Since the format is so wide-spread, considerable research has focused on scheduling DRRTs efficiently and fairly (Rasmussen & Trick, 2008; Trick, 2000; 2002). A problem with DRRTs is that the number of games that each team plays in the competition is directly given by the number of teams in the league: each team in a k-team league plays $2(k-1)$ games. Many smaller leagues may therefore want to transition away from a DRRT to increase the number of games offered to each team, thereby also improving league exposure. For example, the top Danish football league has only 12 teams and uses a triple round-robin tournament to offer a sufficient number of games (Rasmussen, 2008).

Another possibility, the subject of this paper, is to divide the league into n-team divisions, each of which will hold an additional single round-robin tournament. We will focus exclusively on two-division leagues with n-team divisions formed a priori, which we abbreviate as DRRT+2D(n). Elitserien, the top Swedish handball league, plays this format with $n = 7$; the Swedish Innebandy league, SSL, will adopt this format in the upcoming season to “increase the interest for SSL and improve the incomes of the clubs” (Jansson, 2016). The Scottish Premier League, SPL, is also a two-division league, but its divisions are formed based on the outcome of a league round-robin tournament; while SPL does not fall into the DRRT+2D(n) category, its scheduling concerns are likely close to those of DRRT+2D(n) leagues.

Elitserien offers the 14 participating teams a sufficient tournament length of $5n-2=33$ game weeks, leaving time for play-offs, national team activities, and breaks over Christmas and the summer. Compared with DRRTs, modified league formats have received limited attention in the literature, although Trick (2002, Section 5.2) is a notable exception. In this paper, we develop a constraint programming framework for scheduling leagues that combine divisional and round-robin play. Without loss of generality, we will assume that the divisional play precedes the round-robin play, but their relative order makes no difference in principle.

Because DRRT scheduling is already difficult—see, for example, Briskorn (2008) for NP-completeness results—one might assume that scheduling augmented DRRTs is even harder. While this assumption may be true in general, some variations allow for new degrees of freedom or impose new constraints that ultimately make scheduling them easier. One example is the requirement that teams in the DRRT+2D(n) format that meet three times over

* Corresponding author.
E-mail addresses: matsc@sics.se (M. Carlsson), mikaelj@kth.se (M. Johansson), jmlarson@anl.gov (J. Larson).

http://dx.doi.org/10.1016/j.ejor.2016.11.033
0377-2217/

the season (i.e., those in the same division) play in a different venue for consecutive meetings. As we will show, the inclusion of this requirement, denoted the *alternating venue requirement* (AVR), in the general DRRT+2D(n) scheduling problem (outlined in Section 2) ultimately makes constructing a season schedule easier.

In addition to generic DRRT+2D(n) scheduling, we consider in Section 3 the specific constraints imposed by Elitserien. In Section 4 we introduce additional schedule properties that are implied by general DRRT+2D(n) constraints and Elitserien-specific constraints. Constraint models consisting of the general DRRT+2D(n) constraints and the Elitserien-specific requirements are then solved in an integrated manner utilizing the properties from Section 4. This approach differs from the common procedures that reduce the scheduling problem into a set of smaller (and easier) tasks. For example, the schedule-then-break approach of Trick (2000) is widely used in scheduling DRRTs, and a similar approach was adopted by Larson and Johansson (2014) to schedule Elitserien. Their approach first constructs a set of home-away pattern (HAP) sets, not all of which are schedulable with respect to the AVR. Unschedulable HAP sets are removed, and then a tournament template (a tournament containing generic numbers and not actual team names) is generated and ranked according to several factors, including carry-over effects (Russell & Urban, 2006), that are not easily optimized directly. (Elitserien required the construction of such a template for approval by the team owners.) After a template is agreed upon, an integer programming (IP) approach assigns teams to the numbers in the template in a manner that satisfies various constraints (e.g., venue availability or desired derby matches). This decomposed approach of building a HAP set, fixing a template, and then constructing a schedule can result in suboptimal schedules.

Constraint programming (CP) has previously been used successfully to schedule sports leagues. Decomposed CP approaches for scheduling DRRTs were proposed by Henz (2001), Schaerf (1999), and Russell and Urban (2006). CP has also been hybridized with other methods to minimize travel distance in sports tournaments (Benoist, Laburthe, & Rottembourg, 2001; Easton, Nemhauser, & Trick, 2001; Rasmussen & Trick, 2006) and to minimize breaks (consecutive home or away games) (Régin, 2001; van't Hof, Post, & Briskorn, 2010). A hybrid IP/CP approach of Rasmussen and Trick (2007) uses Benders cuts to accelerate a decomposed approach as well. However, case studies solved by *integrated* CP approaches are scarce. We introduce the essential CP terminology in Section 5.

In addition to defining and studying the general DRRT+2D(n) format, this article extends our preliminary work on integrated CP approaches for Elitserien reported in Larson, Johansson, and Carlsson (2014): we explore several other constraint programming models and solution strategies and are able to obtain orders-of-magnitude improvements in computation times compared with our earlier models. Section 6 formulates the general DRRT+2D(n) scheduling problem and the additional Elitserien requirements as constraint models. In Section 7, we analyze the first incorrect choice made when the search tree is explored, allowing us to identify fragments of the CP model that can be strengthened. In addition, we find a reformulation of the cost function that allows us to reduce the time for proving optimality. In Section 8, we show empirically that these techniques yield significantly reduced solution times, solving problems on standard PCs in seconds that previously took days. We expand our study of the initial study of a 14-team league to 16-, 18-, and 20-team cases in order to analyze how our approach scales.

Although we focus on a specific league to highlight how an integrated CP approach can be used to quickly construct a schedule, the approach presented is general enough to be applicable to many other leagues or competitions with divisional and round-robin play and their league-specific requirements. This research is also relevant to competitions looking to easily schedule more matches than is allowed by a DRRT. The proposed DRRT+2D(n) schedule that satisfies the Elitserien requirements has notably few breaks: two teams have no breaks, and the remaining teams have only two breaks in their schedule.

Table 1
General requirements for a DRRT+2D(n).

G1.	The divisions must contain n teams where both division's teams are predefined.
G2.	Each division must hold an SRRT to start the season.
G3.	The divisional SRRT must be followed by a DRRT between the entire league. The DRRT is organized into two SRRTs, where the second SRRT is the mirrored complement of the first: the order is reversed, home games become away games, and vice versa.
G4.	The schedule must have a minimum number of breaks.
G5.	The divisional SRRT must contain no breaks.
G6.	At no point during the season can the number of home and away games played by any team differ by more than 1.
G7.	All pairs of teams must have consecutive meetings occur at different venues. We refer to this constraint as the alternating venue requirement (AVR).
G8.	Venue unavailabilities should be respected to the highest extent possible.

Notation

We use the following terminology. In a period, each team has a home game, an away game, or a bye, denoted `H`, `A`, or `B`, respectively. A team has a *break* if two consecutive games are either both home games or both away games. Two teams have *complementary schedules* if they never play at home at the same time and never play away at the same time.

2. DRRT+2D(n) description

We define a general DRRT+2D(n) instance by the schedule requirements stated in Table 1. We believe these requirements are general enough to be applicable to many leagues. Some of the requirements in Table 1 are *seasonal* constraints that can change from year to year: venue availabilities (G8) will differ, as may the teams constituting each division (G1), for example if teams are promoted and relegated. The remaining requirements are *structural* and are independent of the season being scheduled. Note that the stated requirements do not depend on the number of divisions being two; they are generalizable to any number of divisions (assuming the number of teams is a multiple of the number of divisions), although we do not address such issues here. For condition (G4), a team ending the SRRT and starting the DRRT with the same type of game (home or away) is counted as a break. The inclusion of (G5) ensures that the divisional SRRT is unique up to permutation of the teams (Frončer & Meszka, 2005) ; note that `ABA` and `HBH` are considered breaks (even if they occur at the transition to the DRRT) and that if the number of teams in each division is even, each team still has a bye in the divisional SRRT. The condition (G8) invokes an optimization problem: Minimize the violated venue unavailabilities while satisfying the remaining requirements.

3. Elitserien-specific requirements

Elitserien has size $n = 7$ and has structural and seasonal requirements in addition to the general DRRT+2D(n) conditions (G1)–(G8) listed in Table 1. These Elitserien-specific constraints are denoted (E1)–(E3) and are summarized in Table 2.

We note that conditions (E1) and (E2) overlap slightly but convey different information. For a given season, (E2) may contain only one pair of teams in each division that must have complementary schedules. Nevertheless, Elitserien requires in (E1) that

Table 2
Elitserien-specific requirements.

E1.	Each division must have three pairs of complementary schedules.
E2.	Specific pairs of teams in both divisions must be assigned complementary schedules (e.g., teams that come from the same city or share an arena).
E3.	To increase the visibility of handball, the league arranges derbies in specific periods. Elitserien derby constraints consist of a single period and a set of teams, out of which as many matches as possible should be formed. Alternatively, a single team, a single period, and a set of possible opponents are given.

BAHAHAH	A	H	A	H	A	H	A	H	A	H	A	H	A
HBAHAHA	H	A	A	H	A	H	A	H	A	H	A	H	A
AHBAHAH	A	H	H	A	H	A	H	A	H	A	H	A	H
HAHBAHA	H	A	H	A	A	H	A	H	A	H	A	H	A
AHAHBAH	A	H	A	H	H	A	H	A	H	A	H	A	H
HAHAHBA	H	A	H	A	H	A	A	H	A	H	A	H	A
AHAHAHB	A	H	A	H	A	H	H	A	H	A	H	A	H
BHAHAHA	H	A	H	A	H	A	H	A	A	H	A	H	A
ABHAHAH	A	H	A	H	A	H	A	H	H	A	H	A	H
HABHAHA	H	A	H	A	H	A	H	A	H	A	A	H	A
AHABHAH	A	H	A	H	A	H	A	H	A	H	H	A	H
HAHABHA	H	A	H	A	H	A	H	A	H	A	H	A	A
AHAHABH	A	H	A	H	A	H	A	H	A	H	A	H	H
HAHAHAB	H	A	H	A	H	A	H	A	H	A	H	A	H

Fig. 1. Left: Two HAP sets for a 7-team no-break RRT. Right: HAP set satisfying the DRRT+2D(7) requirements for a 14-team, 12-break RRT. Breaks are highlighted. These HAP sets are unique up to permutation of the rows.

BAHAHAHA	H	A	H	A	H	A	H	A	H	A	H	A	H	A	H
BHAHAHAH	A	H	A	H	A	H	A	H	A	H	A	H	A	H	A
AHBAHAHA	H	H	A	H	A	H	A	H	A	H	A	H	A	H	A
HABHAHAH	A	A	H	A	H	A	H	A	H	A	H	A	H	A	H
AHAHBAHA	H	A	H	H	A	H	A	H	A	H	A	H	A	H	A
HAHABHAH	A	H	A	A	H	A	H	A	H	A	H	A	H	A	H
AHAHAHBA	H	A	H	A	H	H	A	H	A	H	A	H	A	H	A
HAHAHABH	A	H	A	H	A	A	H	A	H	A	H	A	H	A	H
ABHAHAHA	H	A	H	A	H	A	H	H	A	H	A	H	A	H	A
HBAHAHAH	A	H	A	H	A	H	A	A	H	A	H	A	H	A	H
AHABHAHA	H	A	H	A	H	A	H	A	H	H	A	H	A	H	A
HAHBAHAH	A	H	A	H	A	H	A	H	A	A	H	A	H	A	H
AHAHABHA	H	A	H	A	H	A	H	A	H	A	H	H	A	H	A
HAHAHBAH	A	H	A	H	A	H	A	H	A	H	A	A	H	A	H
AHAHAHAB	H	A	H	A	H	A	H	A	H	A	H	A	H	H	A
HAHAHAHB	A	H	A	H	A	H	A	H	A	H	A	H	A	A	H

Fig. 2. Left: Two HAP sets for an 8-team no-break RRT. Right: HAP set satisfying the DRRT+2D(8) requirements for a 16-team, 14-break RRT. Breaks are highlighted. These HAP sets are unique up to permutation of the rows.

two pairs in the remaining five teams have complementary schedules as well.

The general DRRT+2D(n) constraints (G1)–(G8) together with (E1)–(E3) summarize the requirements accounted for by the Swedish Handball Federation when scheduling Elitserien. Traditionally all constraints except (G8) (and to some extent (E3)) are considered hard, while the number of respected venue availabilities and satisfied derby requests is treated as an objective that should be maximized. This reflects the desires of a league where competitive fairness is given the highest priority but is also motivated by practical concerns. Historically, Elitserien has determined its schedule by first proposing a tournament template (containing generic numbers instead of team names) that addresses structural constraints: schedule format and fairness in terms of breaks, byes, complementary schedules, and the alternating venue requirement. Teams are then assigned to a number in a manner that best satisfies the league's seasonal constraints—which teams are in which division and the league's collective wishes and availabilities. The latter category includes stadium and referee availabilities, the desire to support various match-ups (such as rivalries), and wishes from the media. Furthermore, Elitserien has significant flexibility with scheduling games: although a venue may be unavailable for the target date of a specific game round, the league allows the teams flexibility to move a game date a few days forward or backward. For example, a game scheduled for Saturday can be played on Friday or Sunday, depending on venue, referee, and team availabilities. Therefore, (G8) is a soft constraint (and a suitable objective).

4. Structural analysis of DRRT+2D(n) and Elitserien requirements

In this article we present an integrated CP approach for generating a schedule that satisfies requirements (G1)–(G7) and that violates a minimum number of venue unavailabilities (requirement (G8)). We also apply the Elitserien-specific requirements (E1)–(E3) to produce a restricted model, and we generalize the model to $n > 7$. As we will see in Sections 6–8, these models can be strengthened significantly if we can explicitly encode some of the restrictions on feasible home-away patterns that are implied by the requirements. Next, we perform such an analysis and identify several structural properties that will be used to improve our models.

DRRT+2D(n) as defined in this paper induces a specific format on the tournament. Namely, for $n = 7$, the home-away pattern (HAP) for the tournament must be constructed by combining the divisional RRT home-away patterns in Fig. 1 (left) with two copies of a full-season RRT home-away pattern in Fig. 1 (right) without introducing additional breaks. Each team's SRRT HAP is described by a row from Fig. 1 (left). This is completed to a full-season HAP by taking a row from Fig. 1 (right) and appending it plus a reflected complement. In other words, if the row from Fig. 1 (right) ends AHH, then the team's next games will be AAH. The schedule will also mirror this pattern: if team 1 ends its first half of the DRRT playing at team 2's venue, they will host team 2 during its next game. Generally, for odd n, the pattern resembles Fig. 1 as far as placement of byes and breaks is concerned. For even n, the pattern is slightly different and resembles Fig. 2.

Each team's schedule can be considered as three parts: Part I, which is the SRRT, ending in period n; Part II, which is the first half of the DRRT, ending in period $3n - 1$; and Part III, which is the second half of the DRRT, ending in period $5n - 2$. The manner in which the schedule is built depends on whether n is odd or even.

For odd n, the Part I HAPs for Division 1 must be a permutation of the rows of Fig. 1 (left top), and the Part I HAPs for Division 2 must be a permutation of the rows of Fig. 1 (left bottom), or vice versa. Part II must be a permutation of Fig. 1 (right).

For even n, the Part I HAPs for Division 1 must be a permutation of the rows of Fig. 2 (left top or bottom), and the same holds for Division 2. In other words, in contrast to the odd n case, both divisions can use the same HAP set in Part I. Part II must be a permutation of Fig. 2 (right).

Reflecting and taking the complement of Part II to form Part III (and satisfy (G3)) force teams to play the same team in period $3n - 1$ as they do in period $3n$ (at the opposite venue). This schedule could be undesirable, depending on the league, but it is a nonissue for Elitserien. Part II ends before Christmas, allowing for a month-long break for Champions League competitions before Part III starts at the beginning of February.

A number of properties of the HAP set that satisfies the DRRT+2D(n) requirements are useful in developing efficient implied and symmetry-breaking constraints.

- PG1. (G6) implies that breaks can occur only in odd periods of Part II.
- PG2. If n is odd, then in each division, one bye occurs in each period of Part I.
- PG3. If n is odd and the Part I byes are placed as in Fig. 1, then the first row of Part I is complementary to the first column of Part I for both divisions.

Property (PG1) is implied by (G6); properties (PG2) and (PG3) are a direct result of the HAP underlying the unique, no-break divisional SRRT.

The following useful properties can be derived if the Elitserien-specific complementary schedule requirements are imposed. They are given here for $n = 7$, but they can be generalized for any odd n:

- PE1. The three pairs of complementary schedules per division required by (E1) must have breaks that are pairwise aligned, as in Fig. 1.
- PE2. Two HAPs can be complementary only if the byes occur in adjacent periods of Part I (Larson and Johansson, 2014, Proposition 3.3). Visual inspection of Fig. 1 shows that two nonadjacent sequences are noncomplementary in at least one of the periods 1 through 8.
- PE3. If the Part I byes are placed as in Fig. 1, the required three pairs of complementary schedules must include teams 2, 4, and 6 of the given division.
- PE4. By inspecting the known 104 distinct, feasible HAP sets that exist for Elitserien, the two rows with no break must be placed in different divisions, in one of the following ways:

First division	**Second division**
row 1 or 5	row 10 or 14
row 3 or 7	row 8 or 12

Property (PE3) follows directly from the fact that there are only four ways to form three complementary pairs for the HAP rows: $(1+2, 3+4, 5+6)$; $(1+2, 3+4, 6+7)$; $(1+2, 4+5, 6+7)$; and $(2+3, 4+5, 6+7)$. For more details on the distinct 104 Elitserien HAP sets utilized for (PE4), see the work of Larson and Johansson (2014).

For even n, we derive the following property instead:

- PE5. Each pair of complementary schedules must have aligned byes and must have either breaks that are aligned or no breaks, as in Fig. 2.

5. Constraint programming

Having defined the Elitserien schedule requirements, we now review the CP terminology used in this paper. For a deeper introduction to the state of the art of CP, see the work of Rossi, van Beek, and Walsh (2006).

A *constraint satisfaction problem* (CSP) consists of a set of variables

$$X = \{x_1, \ldots, x_k\},$$

where each variable $x_i \in X$ has an associated finite domain $D(x_i) \subset \mathbb{Z}$ and a collection of *constraints*. Declaratively, each constraint is a relation—a set of tuples—over some set of variables. Operationally, each constraint is implemented by a *filtering algorithm* that endeavors to delete any domain values that are not supported by the relation. This requires the domains to be implemented by some data structure that supports such delete operations.

A *solution* to a CSP is an assignment of a value $d_i \in D(x_i)$ to each $x_i \in X$, such that all the constraints are satisfied. Often, we wish to find a solution to a CSP that minimizes or maximizes some function. A *constraint optimization problem* (COP) is a CSP together with an *objective function*

$$f : D(x_1) \times \cdots \times D(x_k) \mapsto \mathbb{Z}.$$

An *optimal solution* to a COP is a solution that optimizes f. A *CP model* of a given satisfaction (optimization) problem is a CSP (COP) with variables, constraints, and optionally an objective function, encoding the problem.

The basic constraint-solving technique is tree search combined with *propagation*, the execution of all filtering algorithms to a fixed point (i.e., until none of the filtering algorithms remove any more domain values), after which three possibilities exist. If some domain has become empty, then the search has encountered a *failure*. If all domain have become singletons, then all variables have been fixed, and a *solution* has been found. Otherwise, a variable x is selected, and two new tree branches are spawned, corresponding to mutually exclusive assumptions on x. A popular variable-choice strategy is *first-fail*, which consists in selecting x such that $|D(x)|$ is minimal. A common branching strategy is to explore $x = \min(D(x))$ vs. $x \neq \min(D(x))$. For optimization problems, this basic tree search is replaced by branch-and-bound search.

When propagation has completed, some domain may contain values that are consistent with every individual constraint, while being inconsistent with their conjunction. For example, assume the constraints $x_1 \neq x_2 \neq x_3 \neq x_1$ with domains $D(x_1) = \{1, 2, 3\}, D(x_2) = D(x_3) = \{2, 3\}$. Assume further that the search has made the assumption $x_1 > 1$. After propagation, we have $D(x_1) = D(x_2) = D(x_3) = \{2, 3\}$, which is consistent with each individual "$\neq$ constraint" but inconsistent with their conjunction, because three distinct integers are needed in any solution. This phenomenon, called *missing propagation*, increases the risk of making bad choices when searching the decision tree leading to dead ends—an inconsistent (or infeasible) set of domain values. Missing propagation is a sign that the CP model is too weak. In the given example, this can be remedied by replacing the conjunction by the global constraint AllDifferent($[x_1, x_2, x_3]$) (defined in Appendix A), whose filtering algorithm can reason globally over the conjunction. Another technique is to add *implied constraints* (Smith, 2006). From a declarative point of view, they are completely redundant and do not remove any solutions. Operationally, however, they may propagate more strongly than the original constraints; that is, they may allow more inconsistent domain values to be deleted.

Suppose now that there exists a mapping

$$m : D(x_1) \times \cdots \times D(x_k) \mapsto D(x_1) \times \cdots \times D(x_k)$$

so that for every solution s to a COP, m maps s to another solution s' such that $f(s) = f(s')$. From an optimization perspective, we are interested in only one optimal solution; hence, eliminating s or s' (but not both!) from the solution space is desirable. A *symmetry-breaking constraint with respect to m* (Gent, Petrie, & Puget, 2006) is a constraint that admits exactly one of s and $m(s)$, for all solutions s. A *channeling constraint* expresses a functional relation that shows up when the CP model contains multiple arrays of variables that represent different aspects of the same information. A *global constraint* (van Hoeve & Katriel, 2006) is a constraint that captures a relation among a nonfixed number of variables. Typically, a global constraint is shorthand for a frequently recurring pattern and greatly simplifies the modeling task. Even though a global constraint can be decomposed into a logical formula over simpler constraints, such decompositions can have a bad space complexity.

Further, for many global constraints, a low-complexity filtering algorithm is known that filters the constraint more effectively than does a naive decomposition. In fact, a global constraint can have multiple, alternative filtering algorithms, trading effectiveness for complexity, or even none at all, relying on decomposition. We list and define the global constraints used in this article in Appendix A. A much larger set of constraints is proposed in the Global Constraint Catalog (Beldiceanu, Carlsson, Demassey, & Petit, 2007).

A CP model can be solved by using one of many existing programming interfaces for entering and executing such models. In this study, we use MiniZinc (Nethercote et al., 2007), a modeling language with a syntax that is close to mathematical notation. MiniZinc models are compiled to a low-level language (FlatZinc) that can be interpreted by multiple back-end solvers, each with a different repertoire of algorithms for filtering, search, learning, and so forth. The compilation of global constraints is thus back-end specific. We selected the Chuffed (Chu, de la Banda, & Stuckey, 2010) back end, which in addition to having a rich repertoire of filtering algorithms is a lazy clause generation (Ohrimenko, Stuckey, & Codish, 2007) solver with nogood learning and VSIDS search (Moskewicz, Madigan, Zhao, Zhang, & Malik, 2001), features that are crucial to the performance or our approach. In fact, Chuffed is best described as a hybrid CP-SAT solver and contains a great deal of modern SAT solver technology. To discover cases of missing propagation, we also used the Gecode back end (Schulte & Tack, 2014) and in particular its graphical tool (Gist), which allows the modeler to inspect the search tree.

6. Constraint programming models

We now describe in detail the integrated CP model of the scheduling problem. We first define the model variables and then present the essential constraints to ensure that the resulting schedule will satisfy the DRRT+2D(n) requirements (G1)–(G8) from Section 2. Next, we present the additional constraints required by the specific Elitserien requirements (E1)–(E3) from Section 3. We then identify implied and symmetry-breaking constraints using the properties (PG1)–(PG3) and (PE1)–(PE4) from Section 4, which greatly reduce the search effort. The resulting CP models were encoded in MiniZinc 2.0 and executed with Chuffed as the back end. Full details of our experiments are given in Section 8.

6.1. Problem variables

Constraint (G3) implies that we need to define variables for only Parts I and II because Part III is the mirrored complement of Part II. To satisfy (G2)–(G3), let $\mathcal{T} = \{1, \ldots, 2n\}$ denote the set of teams, $\mathcal{P} = \{1, \ldots, 3n-1\}$ denote the set of periods in Parts I and II, $t \in \mathcal{T}$ denote a team, and $p \in \mathcal{P}$ denote a period. The tournament template corresponds to the array of variables $T[t,p] \in \{-2n, \ldots, 2n\}$, where $T[t,p] < 0$ if a team t plays away in period p, $T[t,p] > 0$ if it plays at home, and $T[t,p] = 0$ if it has a bye. The HAP set corresponds to the array of variables $H[t,p] \in \{\mathtt{A}, \mathtt{B}, \mathtt{H}\}$. The opponent of team t in period p is contained in the array $O[t,p] \in \mathcal{T}$, where $O[t,p] = t$ if and only if it has a bye in that period. Let $\mathcal{B}$ denote the set of periods in which breaks can occur, plus the integer 0 denoting no break. We have from (G6) that $\mathcal{B} = \{0\} \cup \{i \mid n < i < 3n-1 \wedge i \text{ is odd}\}$. We use an array $B[t] \in \mathcal{B}$ to represent the period in which the break for team t occurs, or 0 if team t has no break in its schedule.

Henz, Müller, and Thiel (2004) show that if the CP model uses opponent variables, as ours does, then an SRRT can be codified by two types of constraints; the constraints' filtering algorithms are crucial to performance. First, every period consists of a matching (or one-factor) of the teams, captured by (5). Second, the complete set of opponents for a given team i is the entire set of teams without team i, captured by (6). Alternatives to opponent variables are discussed by Perron (2005).

In order to deal with (G1), specific *team names* must be substituted for team *row numbers*. This substitution requires a level of indirection in the form of another array R that maps team name u to an integer $t = R[u]$, that is, the corresponding row number. By restricting the domains of the array elements, division membership is enforced.

6.2. DRRT+2D(n) constraints

To define constraints satisfying (G1)–(G8), we need some channeling constraints to relate the T, O, H, and B arrays. The T array is channeled to the O and H arrays by

$$T[t,p] = \begin{cases} -O[t,p], & \text{if } H[t,p] = \mathtt{A} \\ O[t,p], & \text{if } H[t,p] = \mathtt{H}, \forall t, \forall p. \\ 0, & \text{if } H[t,p] = \mathtt{B} \end{cases} \tag{1}$$

The definition of a break and the channeling between the B and H arrays is captured by the constraint

$$B[t] = \sum_{p \in \mathcal{B}\setminus\{0\}} p \times (H[t,p] = H[t,p+1]), \forall t. \tag{2}$$

The channeling between the O and H arrays is captured by the constraint

$$O[t,p] = t \Leftrightarrow H[t,p] = \mathtt{B}, \quad \forall t, \quad \forall p. \tag{3}$$

The possible HAP for any team is constrained by (G4)–(G6) and by the fact that we know the set of sequences that must make up a HAP set satisfying the DRRT+2D(n) requirements; see Fig. 1. This is easily captured by a regular expression e. The corresponding finite automaton is shown in Fig. 3; and the corresponding Regular constraint from Appendix A,

$$\textsc{Regular}([H[t,p] \mid p \in \mathcal{P}], e), \quad \forall t, \tag{4}$$

is imposed on every row of H.

As mentioned previously, every period must consist of a matching of teams; that is,

$$O[O[t,p],p] = t, \quad \forall t, \quad \forall p, \tag{5}$$

which can be encoded by

$$\textsc{Inverse}([O[t,p] \mid t \in \mathcal{T}], [O[t,p] \mid t \in \mathcal{T}]), \quad \forall p.$$

This use of Inverse in fact emulates

$$\textsc{SymmetricAlldifferent}([O[t,p] \mid t \in \mathcal{T}]).$$

The latter constraint was motivated by sports scheduling applications. Unfortunately, its native filtering algorithm is rare among CP solvers and is not available in the MiniZinc back ends that we used. Therefore, we have no data on the amount of improvement we might get by using that algorithm.

Each team must meet every other team in its division during Part I and meet every team in Part II. This condition is easily expressed with Alldifferent:

$$\begin{aligned} &\textsc{Alldifferent}([O[t,p] \mid p \in \{1, \ldots, n\}]) \wedge \\ &\textsc{Alldifferent}([O[t,p] \mid p \in \{n+1, \ldots, 3n-1\}]), \quad \forall t. \end{aligned} \tag{6}$$

Also, home and away must match for every team and its opponent, everywhere:

$$\begin{pmatrix} (H[t,p] = \mathtt{A} \wedge H[O[t,p],p] = \mathtt{H}) \vee \\ (H[t,p] = \mathtt{B} \wedge H[O[t,p],p] = \mathtt{B}) \vee \\ (H[t,p] = \mathtt{H} \wedge H[O[t,p],p] = \mathtt{A}) \end{pmatrix}, \quad \forall t, \forall p. \tag{7}$$

To encode (G7), we note that it is satisfied if and only if every row of the tournament template contains distinct nonzero values.

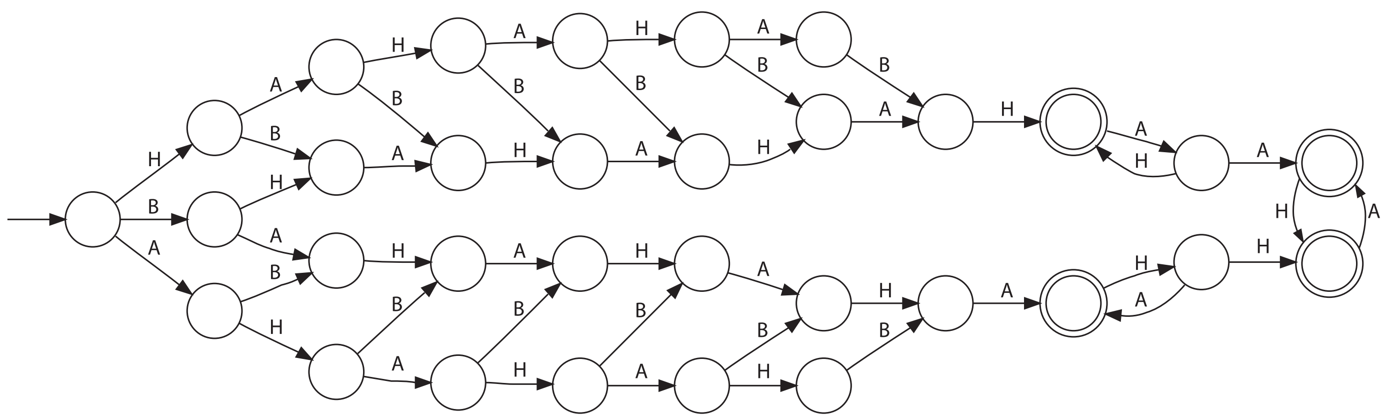


Fig. 3. Finite automaton for valid HAP for $n = 7$. Accepting states are indicated by double circles. When restricted to sequences of length $3n - 1 = 20$, it accepts the combinations of rows in Fig. 1.

Because every team has exactly one bye, this can be enforced by the constraint

$$\textsc{AllDifferent}([T[t,p] \mid p \in \mathcal{P}]), \forall t. \tag{8}$$

As discussed previously, venue unavailabilities (G8) are soft constraints that turn the scheduling problem into an optimization problem. If the preferences are stated as an array,

$$N[u,p] = \begin{cases} 1, & \text{if team } u \text{ prefers not to play at} \\ & \quad \text{home during period } p \\ 0, & \text{otherwise,} \end{cases}$$

then the cost function for all three parts of the schedule is

$$cost = \sum_{\substack{u \in \mathcal{T}, \\ p \in \mathcal{P}}} N[u,p] \times (H[R[u],p] = \mathtt{H}) + \sum_{\substack{u \in \mathcal{T}, \\ p \in \{3n,\dots,5n-2\}}} N[u,p] \times (H[R[u], 6n-1-p] = \mathtt{A}).$$

6.3. Elitserien-specific constraints

We now declare constraints to encode requirements (E1)–(E3). Let $i \oplus j$ denote the fact that teams i and j have complementary schedules. That is,

$$i \oplus j \Leftrightarrow B[i] = B[j] \wedge (H[i,p] \neq H[j,p], \forall p \in \mathcal{P}).$$

Then (E1) can be encoded by

$$\exists \{a,b,c,d,e,f\} \subset \{1,\dots,n\} \text{ such that } (a \oplus b) \wedge (c \oplus d) \wedge (e \oplus f) \text{ and}$$
$$\exists \{a,b,c,d,e,f\} \subset \{n+1,\dots,2n\} \text{ such that } (a \oplus b) \wedge (c \oplus d) \wedge (e \oplus f). \tag{10}$$

Requirement (E2) says that a given set $\mathcal{C}$ of pairs (u, u') of named teams must have complementary schedules:

$$R[u] \oplus R[u'], \forall (u,u') \in \mathcal{C}. \tag{11}$$

Elitserien derby constraints (E3) take one of two forms. The first consists of a period p and a set $\mathcal{Q}$ of four named teams, from which two pairs of playing teams must be formed. This is described by

$$O[R[i],p] \in \{R[j] \mid j \in \mathcal{Q}\}, \quad \forall i \in \mathcal{Q}. \tag{12}$$

Alternatively, a set $\mathcal{T}$ of three named teams is given, two of which must play each other, encoded by

$$(O[R[i],p] = R[j] \vee O[R[i],p] = R[k] \vee O[R[j],p] = R[k]), \quad \text{where } \mathcal{T} = \{i,j,k\}. \tag{13}$$

6.4. Implied constraints

Recall that an implied constraint is logically implied by the essential constraints but may allow more inconsistent domain values to be deleted, thus helping reduce the search effort.

We know from (PG1) and (PE1) that out of the 2n teams, two teams must have a break in each non-last odd period of Part II and two teams must have no break. The requirement

$$\sum_{t \in \mathcal{T}} (B[t] = i) = 2, \forall i \in \mathcal{B}, \tag{14}$$

is efficiently encoded by a GlobalCardinality constraint and was experimentally found to be useful.

For odd n we also have property (PG2), which is useful in itself but which is subsumed by (18), as we shall see later:

$$\textsc{AllDifferent}([p \mid H[t,p] = \mathtt{B}, t \in \{1,\dots,n\}, p \in \{1,\dots,n\}]) \wedge$$
$$\textsc{AllDifferent}([p \mid H[t,p] = \mathtt{B}, t \in \{n+1,\dots,2n\}, p \in \{1,\dots,n\}]). \tag{15}$$

Finally, for odd n, property (PG3) can be imposed as an implied constraint:

$$H[t,1] \neq H[1,t] \wedge H[t+n,1] \neq H[n+1,t], \quad \forall t \in \{2,\dots,n\}, \tag{16}$$

which was determined experimentally to improve propagation, but only marginally.

6.5. Breaking symmetries

Recall that symmetric solutions with equal cost are a source of overhead in optimization search. The following constraints, valid for odd n, help remove many of the symmetries in DRRT+2D(n) scheduling problems.

A first, obvious symmetry is the following. Given a solution, we can construct another solution by swapping home and away everywhere. This symmetry is easily broken by

$$H[2,1] = \mathtt{H} \wedge H[n+2,1] = \mathtt{A}. \tag{17}$$

A second source of symmetry also exists in the model: Given a solution, we can construct another solution by swapping rows (teams) i and j of the same division in the arrays as well as values i and j (positive or negative) in O and T. To break this symmetry, we can fix the bye period for all teams, as in Fig. 1, subsuming (15):

$$\left(\begin{array}{cr} H[t,t] = \mathrm{B} & \wedge \\ O[t,t] = t & \wedge \\ O[t,p] \in \{1,\ldots,n\} \setminus \{t\} \quad \forall p \neq t & \wedge \\ H[t+n,t] = \mathrm{B} & \wedge \\ O[t+n,t] = t+n & \wedge \\ O[t+n,p] \in \{n+1,\ldots,2n\} \setminus \{t+n\} \quad \forall p \neq t & \end{array}\right), \quad \forall t \in \{1,\ldots,n\}. \tag{18}$$

Having fixed the bye periods in such a manner, we can use properties (PE2) and (PE3) to construct a slightly stronger version of (10) that restricts the possible pairing of complementary schedules. Let $\mathcal{E}$ be the set $\{i, i+n \mid 1 < i < n \wedge i \text{ is even}\}$. Then

$$(t-1 \oplus t \vee t \oplus t+1), \quad \forall t \in \mathcal{E}. \tag{19}$$

$$B[t] > 0, \ \forall t \in \mathcal{E}. \tag{20}$$

We note that constraint propagation on the structural constraints and (17)–(18) completely fixes periods 1 to n+2 of the HAP set to the nonpermuted pattern shown in Fig. 1.

If division membership is kept free in (G1), then a third source of symmetry is the fact that the two divisions can be swapped in the template. This symmetry can be broken by lexicographically ordering the break sequences. However, it is not useful in our models, because if used together with the previous two constraints, and because (G8) is based on home vs. away assignments, it may suppress optimal solutions:

$$[B[t] \mid t \in \{1,\ldots,n\}] \leq_{\mathrm{lex}} [B[t] \mid t \in \{n+1,\ldots,2n\}]. \tag{21}$$

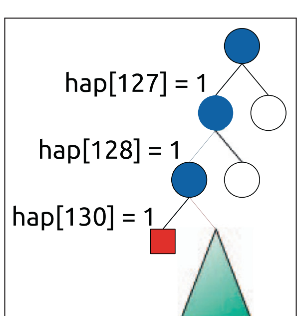
Other symmetry-breaking constraints are discussed by Trick (2000).

7. Strengthening the model

Endeavoring to improve performance, we then tried to strengthen the CP model, identifying cases of missing propagation and adding constraints to prevent such missing propagation. This process was repeated several times, as described in this section. Section 8 contains an evaluation of the evolving sequence of models.

7.1. Missing propagation

A well-known and surprisingly powerful way of finding missing propagation is to focus on the first wrong choice made by the search. This is conveniently spotted with the Gist visualization tool. In the figure to the right, Gist shows the search tree on a chosen benchmark instance with the above model, where square nodes denote dead ends and triangles denote subtrees that can be arbitrarily large. The state at the first mistake corresponds to a partial HAP set of the following form.



$$\begin{array}{|ccccccc|ccccccccccccc|}
\hline
\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\boxed{\mathtt{A}}&\boxed{\mathtt{A}}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\boxed{\mathtt{H}}&\boxed{\mathtt{H}}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\boxed{\mathtt{A}}&\boxed{\mathtt{A}}&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&.&.&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\cline{1-7}
\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&\mathtt{H}&\mathtt{A}&.&.&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&.&.&\mathtt{A}&\mathtt{H}&.&.&.&.&.&.&.\\
\hline
\end{array}$$

Note that the highlighted part clearly violates constraint (14), because we cannot have three breaks in period 9. The problem is that (14) is over the B variables only, whereas (2) links the H and B variables. But (2) is unaware that there can be at most one break per row and therefore cannot yet fix $B[2]$, $B[3]$ and $B[4]$, thus preventing (14) from detecting this dead end. This is a typical case of missing propagation. To remedy it, we replace (2) by

$$B[t] = p \iff H[t,p] = H[t,p+1], \quad \forall t, \quad \forall p \in \mathcal{B} \setminus \{0\}. \tag{22}$$

Upon resolving this issue, this mistake is avoided, and now the first wrong choice corresponds to the partial HAP set.

$$\begin{array}{|ccccccc|ccccccccccccc|}
\hline
\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}\\
\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}\\
\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{B}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}\\
\cline{1-7}
\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\boxed{\mathtt{A}}&\boxed{\mathtt{A}}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}\\
\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\boxed{\mathtt{H}}&\boxed{\mathtt{A}}&\mathtt{H}&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&.&.&.\\
\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&.&.&.&.&.\\
\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{B}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&\mathtt{H}&\mathtt{A}&.&.&.&.&.\\
\hline
\end{array}$$

This time, the highlighted part shows that we have violated (PE3): row 8 has a break in period 15, but row 9 has not. Apparently, (19) is not strong enough to prevent such mistakes. We fix that situation using (PE4) to add

$$\begin{aligned} &B[1] = B[10] = 0 \ \vee \\ &B[1] = B[14] = 0 \ \vee \\ &B[3] = B[8] = 0 \ \vee \\ &B[3] = B[12] = 0 \ \vee \\ &B[5] = B[10] = 0 \ \vee \\ &B[5] = B[14] = 0 \ \vee \\ &B[7] = B[8] = 0 \ \vee \\ &B[7] = B[12] = 0 \end{aligned} \tag{23}$$

and

$$\forall i \in \mathcal{B} \setminus \{0\} \exists t \in \{1,\ldots,n-1,n+1,\ldots,2n-1\} : \\ B[t] = B[t+1] = i. \tag{24}$$

Upon solving the new model, no mistakes are made during HAP construction. Note, however, that an inspection of the search

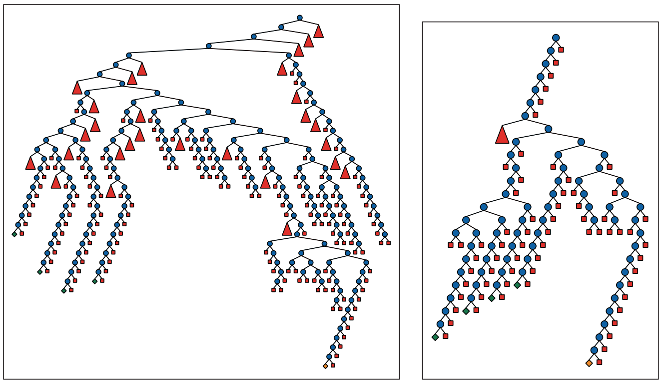


Fig. 4. Search tree visualized by Gist on a given instance with the cost function encoded with (9) (left) vs. (26) and (27) (right). Square nodes denote failures; green (golden) diamonds denote suboptimal (optimal) solutions; triangles denote subtrees that can be arbitrarily large. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

tree reveals that some choices are still made during the search, although we expected that propagation would fix Part I completely. Also, no explicit constraints in the model prevent the construction of an unschedulable HAP set. (Not every HAP set that satisfies the structural constraints can be feasibly assigned teams.) Moreover, during branch-and-bound search, with (9), back propagation from the cost function is extremely weak. These concerns are addressed below.

7.2. Table constraints for HAPs and costs

We now return to the channeling between the H and B arrays, where we saw some unnecessary search. We observe that the finite automaton of Fig. 3 for $n = 7$ and its generalizations to $n = 8$, $n = 9$, and $n = 10$ respectively recognizes 98, 128, 162, and 200 strings respectively. Let a *subrow* be a division-relative row number. For each string s, with symmetry breaking and fixed bye placement, we know

- in which subrow s can occur, as this is decided by the prefix of s; and
- where the bye and any break occur, as these are simple features of s.

In fact, we have a bijection:

$$(\text{string}) \leftrightarrows (\text{subrow}, \text{bye}, \text{break}).$$

Note that for the odd n case, the (absolute) row number is determined by s thanks to the stronger symmetry breaking for odd n. For even n, the subrow is fixed by s, but not the division.

Since the number of recognized strings is not much larger than the size of the automaton, we can introduce an auxiliary variable $Y[t]$ whose value is the bye position of row t and replace (4) by a Table constraint, where each tuple includes the string s, its row, and its break (or 0 if there is none):

$$\textsc{Table}([H[t,1],\ldots,H[t,3n-1],t,Y[t],B[t]],\\ \{[\text{string, row, bye, break}]\}),\quad \forall t. \tag{25}$$

This turns out to subsume not only (4) but also constraints (2), (22), (17), (18), and (16), which consequently can be deleted.

Next, with the help of the Gist tool (see Fig. 4), we noticed that the backtracking activity is concentrated on the right-hand side of the branch-and-bound search tree. Bear in mind that triangles denote collapsed subtrees that can be arbitrarily large. The search effort during the proof of optimality is proportional to the number of nodes traversed by the depth-first search after the optimal solution has been found. In the left-hand tree, the search is dominated by the proof of optimality, which suggested to us that back propagation from the cost function as encoded by (9) might be too weak to be effective. This recurring phenomenon has been observed by other researchers, notably Focacci, Lodi, and Milano (1999).

We therefore tried an alternative formulation of the cost function as a sum over costs per row,

$$cost = \sum_{r\in\mathcal{T}} f(Y[r],B[r],r,R^{-1}[r]), \tag{26}$$

or as a sum of costs per team,

$$cost = \sum_{u\in\mathcal{T}} f(Y[R[u]],B[R[u]],R[u],u), \tag{27}$$

where $f(y, b, r, u)$ can be precomputed as a table for given bye y, break b, row r, and team name u as follows.

1. We first compute the unique *HAP* that is the function of (y, b, r), as explained in Section 7.2.
2. This leaves

$$f(y,b,r,u) = \sum_{p\in\mathcal{P}} N[u,p] \times (HAP[p] = \mathtt{H}) \\ + \sum_{p\in\{3n,\ldots,5n-2\}} N[u,p] \times (HAP[6n-1-p] = \mathtt{A}).$$

It turns out that (26) does not dominate (27), nor vice versa, and that adding both gives the best result. Fig. 4 shows the effect on the shape of the search tree. In the left-hand tree, the search during proof of optimality is 50 nodes, plus 10 collapsed subtrees, several of which containing dozens of nodes. In the right-hand tree, the number is 17 nodes only.

8. Experiments

We endeavored to answer the following research questions about our approach:

- Q1. Do our results for the Elitserien case study generalize to the general DRRT+2D(n) problem?
- Q2. How scalable is our approach?
- Q3. What CP modeling techniques had the best impact on performance?

The CP models[1] were encoded in MiniZinc 2.0[2] and executed with Chuffed,[3] GitHub version of Nov. 4, 2015, as the back end on a quad core 2.8 gigahertz Intel Core i7-860 machine with 8 megabytes of cache per core, running Ubuntu Linux. Chuffed was run with the options `-f -mdd=true` invoking VSIDS search (Moskewicz et al., 2001) and an MDD propagator for Table constraints. The CP model as reported by Larson et al. (2014) corresponds to the SYM curves of Fig. 5. To avoid any ambiguity, we now identify four specific models, which all come in one DRRT+2D(n) variant and one Elitserien-specific variant. In the table below, constraints in brackets are valid for odd n only; the rest are universally valid.

- NOSYM(G) and NOSYM(E) capture structural and seasonal constraints, using the approach to fix team numbers up front, sacrificing symmetry breaking.
- SYM(G) and SYM(E) capture structural and seasonal constraints, using the approach to treat the matching of team names to team numbers as part of the problem.
- STR(G) and STR(E) are strengthened versions of SYM(G) and SYM(E).

1 http://www.sics.se/~matsc/Elitserien.
2 http://www.minizinc.org/.
3 https://github.com/geoffchu/chuffed.

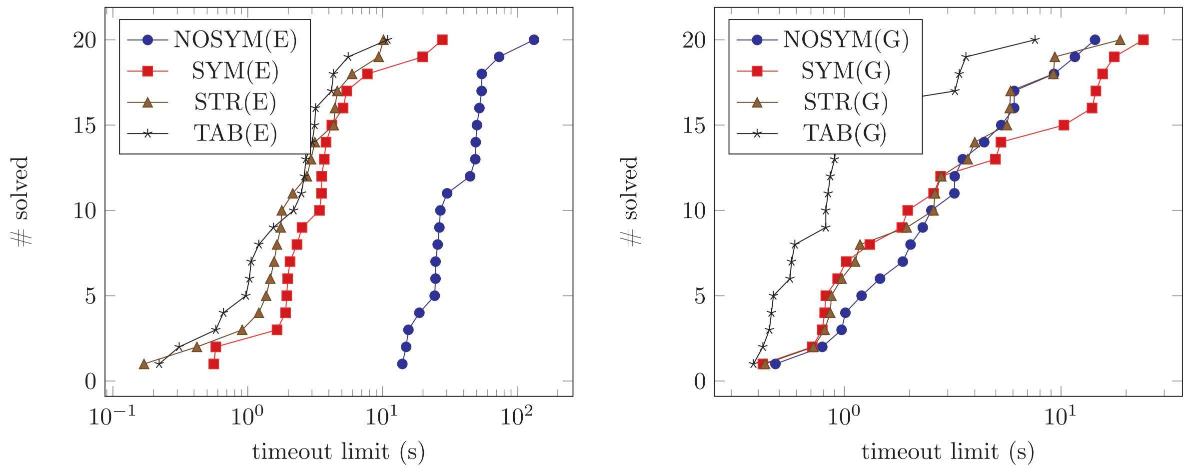


Fig. 5. Number of instances solved to optimality as a function of timeout limit in seconds for DRRT+2D(n) models including Elitserien-specific constraint (left) and excluding them (right).

- TAB(G) and TAB(E) contain Table constraints for HAPs and the cost function.

Model	Constraints
NOSYM(G)	1,3,5,6,7,8,14,2,4,9, [15]
NOSYM(E)	1,3,5,6,7,8,14,11,13,12,2,4,9,10, [15]
SYM(G)	1,3,5,6,7,8,14,2,4,9, [16,17,18]
SYM(E)	1,3,5,6,7,8,14,11,13,12,2,4,9, [16,17,18,19,20,23,24]
STR(G)	1,3,5,6,7,8,14,4,9,22, [16,17,18]
STR(E)	1,3,5,6,7,8,14,11,13,12,4,9,22, [16,17,18,19,20,23,24]
TAB(G)	1,3,5,6,7,8,14,25,26,27
TAB(E)	1,3,5,6,7,8,14,11,13,12, [19,20,23,24], 25,26,27

8.1. Solving the optimization problem

We generated 20 random instances of possible constraints and desires from the leagues because (a) our model has been used only for a single season and we wished to verify that this instance was not an especially easy case to solve and (b) the league requested that, for privacy reasons, we not divulge the true team desires. The structure of the random seasonal constraints was similar to the real ones, however. It involved the following:

- The partitioning of teams into divisions (G1)
- One specific pair of teams in both divisions to be assigned complementary schedules (E2)
- One 3-team intradivision derby set, one 3-team interdivision derby set, and one 4-team interdivision derby set (E3).
- For each team u and period p, u prefers to not play at home during period p with probability 0.05, yielding on average 25 unavailabilities (G8), which is the number of unavailabilities requested by Elitserien teams for the season that was scheduled previously

The minimal, average, and maximal optimal costs (i.e., the number of scheduling conflicts) were 0, 3, and 6, respectively.

We also evaluated two ways of dealing with the requirement (G1):

1. Fixing team numbers, that is, the R array, up front, sacrificing symmetry-breaking, as encoded by model NOSYM.
2. Treating the matching of team names to team numbers as part of the problem, keeping the R array as decision variables, as encoded by the other models. This allowed us to keep the symmetry breaking constraints (16)–(20), which are very effective.

Fig. 5 shows a performance comparison of the models in terms of number of instances solved to optimality as a function of elapsed time. On the left-hand side, the models with the Elitserien-specific constraints were used. On the right-hand side, those constraints were disabled. A general observation is that the Elitserien-specific constraints do not affect the runtimes significantly, thus giving some evidence for an affirmative answer to question (Q1). We also note that no extreme outliers exist among our random instances.

In the Elitserien case, NOSYM is the worst-performing model. This result was unexpected because in models other than NOSYM the R array in effect acts as a level of indirection and inflates the search space, a technique that usually incurs overhead. Evidently the pruning power of the symmetry-breaking constraints outweighs the overhead of the R array, at least if those constraints are effective enough. This partly answers question (Q3).

In both comparisons, TAB is the best-performing model, with the greatest difference for the DRRT+2D(n) comparison. The fact that STR(E) includes (23) and (24) with no counterpart in STR(G) is a possible explanation for the smaller difference between TAB(E) and STR(E) than between TAB(G) and STR(G). Thus the offline processing necessary to compute the extensions of the Table constraints seems to have been a worthwhile investment; this also partly answers question (Q3).

8.2. Scalability

To investigate how the best approach (TAB) scales, we attempted to schedule larger league sizes (up to 10 teams per division) keeping all the requirements, except that (E1) was tightened. For the larger leagues, both divisions must have four pairs of complementary schedules. (It is not possible to schedule a 20-team league in a manner satisfying the requirements in (G1)–(G7) with five pairs of complementary teams in each division.) Property (PE1) was modified accordingly.

For odd division sizes, the home-away pattern is a straightforward extrapolation of the size 7 case; see Fig. 1. In particular, the two divisions must use complementary HAP sets in Part I, and so the structural and symmetry-breaking constraints suffice to completely fix the Part I HAP set.

For even division sizes, the home-away pattern is slightly different. For an 8-team division, the tournament pattern is constructed by combining the divisional RRT home-away patterns in Fig. 2 (left) with the full-league RRT home-away pattern in Fig. 2 (right), plus a mirror image of the second part. As in the 7-team-division case, Part I of Division 1 must be a permutation of Fig. 2 (left top or bottom), and the same holds for Division 2.

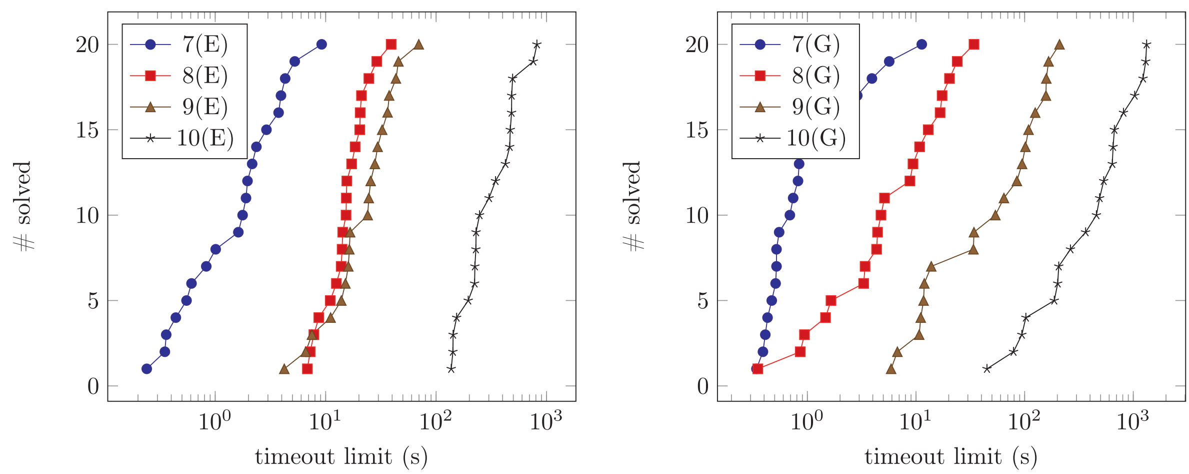


Fig. 6. Number of instances solved to optimality as a function of timeout limit in seconds, for 7, 8, 9, and 10 teams per division, including Elitserien-specific constraint (left) and excluding them (right).

Unlike the 7-team-division case, however, both divisions can use the same HAP set in Part I. Thus, for 8-team divisions the structural and symmetry-breaking constraints do not suffice to completely fix the Part I HAP set to the nonpermuted pattern shown in Fig. 2. Instead, four combinations are possible.

Patterns for even $n > 8$ are straightforward extrapolations of the 8-team pattern.

For even division sizes, the properties (PG2), (PG3), (PE2), (PE3), and (PE4) do not apply, and consequently we cannot use constraints (15)–(24).

From (PE5) we derive the following Elitserien-specific constraint for 8-team divisions,

$$B[2t-1] = B[2t] \quad \forall t \in \{1, \dots, 8\}, \tag{28}$$

and for 10-team divisions,

$$\sum_{t \in \{1,\dots,5\}} (B[2t-1] = B[2t]) \geq 4$$
$$\sum_{t \in \{6,\dots,10\}} (B[2t-1] = B[2t]) \geq 4. \tag{29}$$

For divisions of size 8, 9, and 10, we constructed a version of TAB with the modifications mentioned above. We generated 20 random instances in exactly the same way as for the original case, with the same density of venue unavailabilities, and measured the performance. Fig. 6 compares the performance with Elitserien-specific constraints included (left) and excluded (right). We observe an increase in runtimes of about one order of magnitude per increase in division size, with no instance taking more than 23 CPU-minutes to solve to optimality for 10-team divisions. For the Elitserien-specific runs, we also see a larger increase in runtime from $n = 7$ to $n = 8$ and from $n = 9$ to $n = 10$ than from $n = 8$ to $n = 9$. This is a result of extra implied constraints that stem from properties (PE1)–(PE4), which are valid for odd n only.

The scalability study confirms the previous observation that the Elitserien-specific constraints do not significantly affect the runtimes. This study also strengthens the evidence for an affirmative answer to question (Q1). We also have an answer to question (Q2): the approach easily scales up to at least divisions of 10 teams.

9. Discussion

The integrated CP approach for scheduling the Elitserien is a significant improvement over the decomposed approach by Larson and Johansson (2014). With that approach, we first generated 80,640 HAP sets satisfying (G4)–(G6) but not necessarily schedulable, then applied necessary conditions for schedulability to rule out 87% of the unschedulable HAP sets. An attempt was then made to convert the remaining HAP sets to templates by solving an integer program. The resultant templates were ranked in their carry-over effect to produce a template for the league. This template was then assigned teams with respect to the seasonal requirements (E2)–(E3) and (G1). Testing all HAP sets against the necessary conditions took nearly a day. Since the template was fixed before the seasonal constraints were available, a suboptimal schedule was likely produced. Furthermore, a straightforward application of the approach by Larson and Johansson (2014) to scheduling where a template does not need to be fixed a priori would clearly be inefficient: the 104 schedulable HAP sets admit 5,961,704 templates if constraints (17) and (21) are used, or 23,846,816 templates if they are not. Assuming that it takes 0.1 seconds per template to assign teams to numbers optimally, an optimistic estimate, finding the best schedule would take almost one month.

To exclude the possibility that the Elitserien-specific requirements constrain the problem so much that no conclusions can be drawn for the general DRRT+2D(n) case, we ran all experiments both for the general case and for the Elitserien-specific case. Our results show that the Elitserien-specific requirements do not have a major impact on problem difficulty and that our approach is feasible for the general DRRT+2D(n) case, easily scaling up to league sizes of 10 teams per division.

Our CP model, which integrates the different phases that sports scheduling traditionally decomposes to, shows a dramatic improvement over previous approaches using decomposition and integer programming. Such integrated approaches are rare in the sports scheduling literature. By careful use of implied and symmetry-breaking constraints, as well as a limited amount of off-line processing, we were able to dramatically reduce the time to solution, making CP an attractive technology for producing optimal tournament schedules.

Although we focus on a specific league, the integrated approach presented in this article is general enough to be used by leagues with round-robin and divisional play or competitions looking to play more matches than prescribed by a DRRT without expanding the number of participants.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced

Scientific Computing Research, SciDAC and Applied Mathematics programs under Contract DE-AC02-06CH11357. Mikael Johansson was funded in part by the Swedish Foundation for Strategic Research and the Swedish Science Council. We thank Christian Schulte for his valuable comments. We thank Gail Pieper for her useful language editing.

Appendix A. Global constraint definitions

Following is a list of global constraints used to define the general DRRT+2D(n) and Elitserien-specific CP models.

- AllDifferent($[x_1, \dots, x_k]$) holds if and only if $x_1, \dots, x_k$ are pairwise different. The most popular filtering algorithm is due to (Régin, 1994).
- SymmetricAllDifferent($[x_1, \dots, x_k]$) holds if and only if $\forall i \in \{1, \dots, k\} : x_i \in \{1, \dots, k\}$ and $\forall i \in \{1, \dots, k\}, j \in \{1, \dots, k\} : x_i = j \iff x_j = i$. The same constraint is also known as OneFactor in some papers. The filtering algorithm is due to (Régin, 1999) and was motivated by sports scheduling applications.
- Inverse($[x_1, \dots, x_k], [y_1, \dots, y_k]$) holds if and only if $\forall i \in \{1, \dots, k\} : x_i \in \{1, \dots, k\} \wedge y_i \in \{1, \dots, k\}$ and $\forall i \in \{1, \dots, k\}, j \in \{1, \dots, k\} : x_i = j \iff y_j = i$. The constraint is due to (COSYTEC, 1997).
- GlobalCardinality($[x_1, \dots, x_k], [v_1, \dots, v_n], [c_1, \dots, c_n]$) holds if and only if $[v_1, \dots, v_n]$ are distinct integers, $\forall i \in \{1, \dots, k\} : x_i \in [v_1, \dots, v_n]$, and $\forall j \in \{1, \dots, n\} : c_j = |\{i \in \{1, \dots, k\} \mid x_i = v_j\}|$. The classic filtering algorithm is due to (Régin, 1996).
- Regular($[x_1, \dots, x_k], m$) holds if and only if m is a deterministic finite automaton recognizing a regular language and $[x_1, \dots, x_k]$ is a string that is a member of that regular language (Aho and Ullman, 1994, Chapter 10). The most well-known filtering algorithm is due to (Pesant, 2004).
- Table($[x_1, \dots, x_k], r$) holds if and only if r is a relation, given for example as an explicit list of tuples of values, and $[x_1, \dots, x_k]$ is a tuple that is in the relation. Multiple filtering algorithms have been proposed, for example (Cheng & Yap, 2010; Gent, Jefferson, Miguel, & Nightingale, 2007; Lecoutre, 2011; Lecoutre, Likitvivatanavong, & Yap, 2015; Lecoutre & Szymanek, 2006).

References

Aho, A. V., & Ullman, J. D. (1994). Foundations of computer science. *Principles of Computer Science Series*. W. H. Freeman.

Beldiceanu, N., Carlsson, M., Demassey, S., & Petit, T. (2007). Global constraint catalogue: Past, present and future. *Constraints, 12*(1), 21–62.

Benoist, T., Laburthe, F., & Rottembourg, B. (2001). Lagrange relaxation and constraint programming collaborative schemes for travelling tournament problems. In *CPAIOR: vol. 1* (pp. 15–26).

Briskorn, D. (2008). Sports leagues scheduling. *Lecture Notes in Economics and Mathematical Systems: vol. 603*. Springer. doi:10.1007/978-3-540-75518-0.

Cheng, K. C. K., & Yap, R. H. C. (2010). An MDD-based generalized arc consistency algorithm for positive and negative table constraints and some global constraints. *Constraints, 15*(2), 265–304. doi:10.1007/s10601-009-9087-y.

Chu, G., de la Banda, M. G., & Stuckey, P. J. (2010). Automatically exploiting subproblem equivalence in constraint programming. In A. Lodi, M. Milano, & P. Toth (Eds.), *CPAIOR*. In *LNCS: vol. 6140* (pp. 71–86). Springer. doi:10.1007/978-3-642-13520-0_10.

COSYTEC (1997). CHIP reference manual. Release 5.1.

Easton, K., Nemhauser, G., & Trick, M. (2001). The traveling tournament problem description and benchmarks. In T. Walsh (Ed.), *CP*. In *LNCS: vol. 2239* (pp. 580–584). Springer. doi:10.1007/3-540-45578-7_43.

Focacci, F., Lodi, A., & Milano, M. (1999). Cost-based domain filtering. In J. Jaffar (Ed.), *CP*. In *LNCS: vol. 1713* (pp. 189–203). Springer. doi:10.1007/978-3-540-48085-3_14.

Frončeк, D., & Meszka, A. (2005). Round robin tournaments with one bye and no breaks in home-away patterns are unique. In *Multidisciplinary scheduling: Theory and applications* (pp. 331–340). Springer. doi:10.1007/0-387-27744-7_16.

Gent, I. P., Jefferson, C., Miguel, I., & Nightingale, P. (2007). Data structures for generalised arc consistency for extensional constraints. In *AAAI* (pp. 191–197). AAAI Press.

Gent, I. P., Petrie, K. E., & Puget, J.-F. (2006). Symmetry in constraint programming. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of constraint programming chapter 10*. New York, NY, USA: Elsevier. doi:10.1016/s1574-6526(06)80014-3.

Henz, M. (2001). Scheduling a major college basketball conference—revisited. *Operations Research, 49*(1), 163–168. doi:10.1287/opre.49.1.163.11193.

Henz, M., Müller, T., & Thiel, S. (2004). Global constraints for round robin tournament scheduling. *European Journal of Operational Research, 153*(1), 92–101. doi:10.1016/S0377-2217(03)00101-2.

Jansson, H. (2016). It's done: There will now be more matches in the men's SSL (in Swedish). [Online; posted 2016-03-21]URL: http://innebandymagazinet.se/IBM/artikel.asp?aid=60376.

Larson, J., & Johansson, M. (2014). Constructing schedules for sports leagues with divisional and round-robin tournaments. *Journal of Quantitative Analysis in Sports, 10*(2), 119–129. doi:10.1515/jqas-2013-0090.

Larson, J., Johansson, M., & Carlsson, M. (2014). An integrated constraint programming approach to scheduling sports leagues with divisional and round-robin tournaments. In H. Simonis (Ed.), *CPAIOR*. In *LNCS: vol. 8451* (pp. 144–158). Springer. doi:10.1007/978-3-319-07046-9_11.

Lecoutre, C. (2011). STR2: Optimized simple tabular reduction for table constraints. *Constraints, 16*(4), 341–371. doi:10.1007/s10601-011-9107-6.

Lecoutre, C., Likitvivatanavong, C., & Yap, R. H. C. (2015). STR3: A path-optimal filtering algorithm for table constraints. *Artificial Intelligence, 220*, 1–27. doi:10.1016/j.artint.2014.12.002.

Lecoutre, C., & Szymanek, R. (2006). Generalized arc consistency for positive table constraints. In F. Benhamou (Ed.), *CP*. In *LNCS: vol. 4204* (pp. 284–298). Springer. doi:10.1007/11889205_22.

Moskewicz, M. W., Madigan, C. F., Zhao, Y., Zhang, L., & Malik, S. (2001). Chaff: engineering an efficient SAT solver. In *Design automation conference* (pp. 530–535). ACM. doi:10.1145/378239.379017.

Nethercote, N., Stuckey, P., Becket, R., Brand, S., Duck, G. J., & Tack, G. (2007). MiniZinc: Towards a standard CP modelling language. In C. Bessiere (Ed.), *CP*. In *LNCS: vol. 4741* (pp. 529–543). Springer. doi:10.1007/978-3-540-74970-7_38.

Ohrimenko, O., Stuckey, P., & Codish, M. (2007). Propagation = lazy clause generation. In C. Bessiere (Ed.), *CP*. In *LNCS: vol. 4741* (pp. 544–558). Springer. doi:10.1007/978-3-540-74970-7_39.

Perron, L. (2005). Alternate modeling in sport scheduling. In P. van Beek (Ed.), *CP*. In *LNCS: vol. 3709* (pp. 797–801). Springer. doi:10.1007/11564751_67.

Pesant, G. (2004). A regular language membership constraint for finite sequences of variables. In M. Wallace (Ed.), *CP*. In *LNCS: vol. 3258* (pp. 482–495). Springer. doi:10.1007/978-3-540-30201-8_36.

Rasmussen, R. (2008). Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research, 185*, 795–810. doi:10.1016/j.ejor.2006.12.050.

Rasmussen, R. V., & Trick, M. A. (2006). The timetable constrained distance minimization problem. In J. C. Beck, & B. Smith (Eds.), *CPAIOR*. In *LNCS: vol. 3990* (pp. 167–181). Springer. doi:10.1007/11757375_15.

Rasmussen, R. V., & Trick, M. A. (2007). A Benders approach for the constrained minimum break problem. *European Journal of Operational Research, 177*(1), 198–213. doi:10.1016/j.ejor.2005.10.063.

Rasmussen, R. V., & Trick, M. A. (2008). Round robin scheduling—a survey. *European Journal of Operational Research, 188*(3), 617–636. doi:10.1016/j.ejor.2007.05.046.

Régin, J.-C. (1994). A filtering algorithm for constraints of difference in CSPs. In *AAAI* (pp. 362–367). AAAI Press.

Régin, J.-C. (1996). Generalized arc consistency for global cardinality constraint. In *AAAI* (pp. 209–215). AAAI Press.

Régin, J.-C. (1999). The symmetric alldiff constraint. In T. Dean (Ed.), *IJCAI* (pp. 420–425). Morgan Kaufmann. http://ijcai.org/Proceedings/99-1/Papers/061.pdf.

Régin, J.-C. (2001). Minimization of the number of breaks in sports scheduling problems using constraint programming. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 57*, 115–130.

Rossi, F., van Beek, P., & Walsh, T. (Eds.). (2006). *Handbook of constraint programming*. New York, NY, USA: Elsevier.

Russell, R. A., & Urban, T. L. (2006). A constraint programming approach to the multiple-venue, sport-scheduling problem. *Computers & Operations Research, 33*(7), 1895–1906. doi:10.1016/j.cor.2004.09.029.

Schaerf, A. (1999). Scheduling sport tournaments using constraint logic programming. *Constraints, 4*(1), 43–65. doi:10.1023/A:1009845710839.

Schulte, C., & Tack, G. (2014). *Modeling and programming with Gecode*. URL: http://www.gecode.org/doc-latest/MPG.pdf.

Smith, B. (2006). Modelling. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of constraint programming chapter 11*. New York, NY, USA: Elsevier. doi:10.1016/s1574-6526(06)80015-5.

Trick, M. A. (2000). A schedule-then-break approach to sports timetabling. In E. K. Burke, & W. Erben (Eds.), *PATAT*. In *LNCS: vol. 2079* (pp. 242–253). Springer. doi:10.1007/3-540-44629-X_15.

Trick, M. A. (2002). Integer and constraint programming approaches for round robin tournament scheduling. In E. K. Burke, & P. D. Causmaecker (Eds.), *PATAT*. In *LNCS: vol. 2740* (pp. 63–77). Springer. doi:10.1007/978-3-540-45157-0_4.

van Hoeve, W.-J., & Katriel, I. (2006). Global constraints. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of constraint programming chapter 6*. New York, NY, USA: Elsevier. doi:10.1016/s1574-6526(06)80010-6.

van't Hof, P., Post, G., & Briskorn, D. (2010). Constructing fair round-robin tournaments with a minimum number of breaks. *Operations Research Letters, 38*, 592–596. doi:10.1016/j.orl.2010.08.008.