Annals of Operations Research (2019) 275:715–730
https://doi.org/10.1007/s10479-018-3013-x

ORIGINAL RESEARCH

Scheduling a non-professional indoor football league: a tabu search based approach

David Van Bulck[1] · Dries R. Goossens[1] · Frits C. R. Spieksma[2]

Published online: 24 August 2018

Abstract
This paper deals with a real-life scheduling problem of a non-professional indoor football league. The goal is to develop a schedule for a time-relaxed, double round-robin tournament which avoids close successions of games involving the same team in a limited period of time. This scheduling problem is interesting, because games are not planned in rounds. Instead, each team provides time slots in which they can play a home game, and time slots in which they cannot play at all. We present an integer programming formulation and a heuristic based on tabu search. The core component of this algorithm consists of solving a transportation problem, which schedules (or reschedules) all home games of a team. Our heuristic generates schedules with a quality comparable to those found with IP solvers, however with considerably less computational effort. These schedules were approved by the league organizers, and used in practice for the seasons 2009–2010 till 2016–2017.

Keywords Time-relaxed scheduling · Non-professional · Indoor football · Tabu search

1 Introduction

Despite the huge number of people that compete in sports at a recreational level, academic interest for scheduling non-professional leagues is rather limited. This is in contrast with the large strand of literature which focuses on professional leagues [see e.g. Goossens and Spieksma (2012); for a complete overview we refer to Kendall et al. (2010) and Knust (2017)]. Perhaps, this observation can be explained by the fact that non-professional leagues tend to

A preliminary version of this paper appeared in the Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling. PATAT 2014, pp. 167–178.

✉ David Van Bulck
david.vanbulck@ugent.be

Dries R. Goossens
dries.goossens@ugent.be

Frits C. R. Spieksma
f.c.r.spieksma@tue.nl

[1] Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium

[2] Department of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands

attract less public attention. Hence, scheduling constraints coming from broadcasting rights, security forces, public transport, media and fans usually are non-existing in non-professional competitions. This does not imply however that non-professional scheduling problems are less challenging than their professional counterparts. Indeed, stadium or venue availability is in general more limited, because the team's venue tends to be shared with other teams or sports disciplines. Moreover, practical considerations for the players are far more important, since they have other activities (e.g. family, work) as well.

In this paper, we discuss a sports scheduling problem that originates from the "Liefhebbers Zaalvoetbal Cup (LZV Cup)", a non-professional indoor football league founded in 2002. This league currently involves 477 teams, grouped in 20 regions in Flanders of which the main one has 11 independent divisions and is situated in the vicinity of Leuven (Belgium). In a division, each team plays against each other team twice. The league focuses on teams that consist of friends, is open to all ages, and considers fair play of paramount importance. The games are played without referees, since, according to the organizers, "referees are expensive, make mistakes, and invite players to explore the borders of sportsmanship" (see also www.lzvcup.be [in Dutch]).

Three aspects make this non-professional indoor football scheduling problem different from most problems dealt with in the sport scheduling literature. Firstly, each division plays according to a so-called time-relaxed schedule, which means that the season has (many) more time slots than there are games per team. This is different from time-constrained scheduling, where the number of available time slots equals the minimum number required to schedule all games. Secondly, it is important to balance the rest days between games involving the same team (Suksompong 2016). The basic idea behind this is that most players prefer not to devote a whole weekend to their sport. Moreover, games packed together could also lead to injuries. Thirdly, the league organizers are not worried about breaks, i.e. series of consecutive home games (or away games). The main reason is that the home advantage is quite limited because there are usually few spectators. In professional leagues, in contrast, alternation of home and away games is usually the most important constraint (Goossens and Spieksma 2011). This makes our problem different since the known methods for (professional) sport scheduling—such as first-break-then-schedule (Nemhauser and Trick 1998; Goossens and Spieksma 2012), or canonical one-factorization (de Werra 1981; Goossens and Spieksma 2011)—typically assume a time-constrained schedule where break-minimization is paramount.

In Sect. 2, we give a formal description of the problem, followed by an overview of the literature on related problems in Sect. 3. In Sect. 4, we provide an integer programming formulation. In Sect. 5, we develop a tabu search method, where we search through the neighborhood of a solution by solving a transportation problem. In Sect. 6, we use this heuristic to solve real-life instances, and we compare the outcomes of the heuristic with optimal solutions of the integer program solved by the GUROBI optimizer.

2 Problem description

In this section, we provide a formal problem description, and we introduce the notation used in the remainder of this paper. The teams in the indoor football league are grouped into independent divisions based on their strength. In each division, a double round-robin tournament is played, i.e. each team meets each other team twice (once at its home venue, and once at the opponent's venue). A division has a set of teams $T = \{1, 2, \ldots, n\}$, and a

set of time slots $S = \{1, 2, \ldots, |S|\}$, ranging from the first day of the season till the last. All games should be played within this time frame.

Each team $i \in T$ provides a list of time slots $H_i \subseteq S$ for which their home venue is available. Home games for a team can only be scheduled in time slots from this list. Obviously, if each game is to be scheduled, each team should provide at least as many time slots as it has opponents, i.e. $|H_i| \geqslant n - 1$. This list is called the *home game set*. Some teams may have a time slot on the same weekday every other week; other teams may have a more irregular home game set. Each team $i \in T$ can also provide a list of time slots $A_i \subseteq S$ in which it does not want to play any game; we call this list the *forbidden game set*. Teams can use this list to avoid games during inconvenient periods such as Christmas and New Year, holidays, or examination periods. The forbidden game set implies that in all time slots not in the list, the team is able to play an away game. Since home time slots that are part of the forbidden game sets can never be used, we assume that $H_i \cap A_i = \emptyset$ for each $i \in T$. Besides, a team is not allowed to play twice in the same time slot, or more than twice in a period of R_{max} time slots. Finally, to increase the attractiveness of the schedule, there should be at least m time slots between two games featuring the same pair of teams. Notice that it is allowed to meet an opponent for the second time, before all other opponents have been faced once.

In summary, a feasible solution for this problem consists of an assignment of games to time slots such that:

(C1) each team plays a home game against each other team at most once,
(C2) home team availability H_i $(i \in T)$ is respected,
(C3) away team unavailability A_i $(i \in T)$ is respected,
(C4) each team plays at most one game per time slot,
(C5) each team plays at most 2 games in a period of R_{max} time slots, and
(C6) there are at least m time slots between two games with the same pair of teams.

Constraints *(C1)* express that it is possible not to schedule a game; this will result in a high cost for the objective function. By not stipulating that each team must play a home game against each other team, we guarantee feasibility of each instance (some instances feature teams that do not provide enough time slots for which their home venue is available, i.e., where $|H_i| < n - 1$). In practice, if a game cannot be scheduled, the league organizers leave it to the home team to find a suitable date and location to play the game (if the home team fails to find a suitable time slot, they lose the game). In addition, the goal is to develop a schedule in which each team has a balanced spread of their games over the season. More in particular, teams wish to avoid having two games in a period of R_{max} time slots or less. We use p_r to denote the non-negative penalty incurred for every pair of consecutive games played by a team within a period of $r \in R = \{2, 3, \ldots, R_{max}\}$ time slots. The penalty rates are defined by the league organizers and decrease with the number of time slots between two consecutive games. If a team has $R_{max} - 1$ full time slots or more without a game, we assume that the league organizers no longer care, and consider any number greater than or equal to $R_{max} - 1$ as equally adequate.

3 Related work

A large number of sports scheduling papers deal with professional leagues, e.g. the Chilean soccer leagues (Alarcón et al. 2017), the Belgian soccer league (Goossens and Spieksma 2012), the Australian football league (Kyngäs et al. 2017), and the German basketball league (Westphal 2014). However, for reasons explained in the introduction, the methods used in

the previous papers are not suitable for our problem. Nevertheless, a few problems similar to ours are addressed in the literature.

Costa (1995) proposes a method to generate a time-relaxed schedule for the professional North American National Hockey League (NHL). In addition to other constraints, a team should not play games in three time slots straight, nor should it play more than three games in five consecutive time slots. What is more, each team provides a list with home time slots in which its venue is available, and a list with time slots in which it cannot play at all. Unlike our problem, however, breaks are important and total distance traveled should be minimized. To tackle this problem, Costa (1995) proposes a genetic algorithm in which the mutation phase is replaced by a tabu based search. Easton et al. (2001) isolated the problem of constructing a distance-minimal time-constrained double round-robin tournament while respecting home-away pattern constraints into what is known as the traveling tournament problem (TTP). Later, Bao and Trick (2010) proposed a time-relaxed variant of this problem. Although many contributions have been made for the time-constrained TTP, only two solution methods have been proposed for the time-relaxed TTP. The first is a branch-and-bound procedure that employs metaheuristics to quickly improve upper bounds, and dynamic programming to provide tight lower bounds (Brandão and Pedroso 2014). The second methodology is heuristic and uses an artificial immune algorithm equipped with new move operators to deal with the time-relaxed structure of the tournament (Pérez-Cáceres and Riff 2015).

Suksompong (2016) studies so-called asynchronous round-robin tournaments, where no two games are allowed to take place at the same time. This additional constraint allows spectators to follow all the games live; this occurs, for instance, when there is only one stadium to host all games. To increase the fairness of such tournament, it is important for all teams to have a similar recovery period between two consecutive games. As such, the order in which the games are scheduled, is an important factor in increasing the fairness of the schedule. Likewise, it is desirable that at any moment the number of games played per team is more or less the same. Suksompong (2016) develops three new measures that assess the above two criteria. The canonical schedule turns out to perform well on these three measures when the number of teams is even, but not so well when the number of teams is odd.

Della Croce et al. (1999) consider a tennis tournament in which the players are partitioned into a series of groups. In each group, a time-constrained single round-robin tournament is played such that each participant plays at most once a round; a round consists of several time slots within the same week. The series, however, cannot be scheduled independently as they all share the same venues, for which a weekly venue availability is predefined. Similar to our setting the authors consider player availability, breaks are unimportant, and the objective is to maximize the number of feasible games. To solve the problem, a two-step solution procedure is proposed: first generate the rounds of the tournament without considering venue and player availability constraints, then assign the rounds to the weeks (Della Croce et al. 1999).

Schönberger et al. (2000) discuss a sports scheduling problem faced by a regional non-professional table-tennis association in Germany, involving more than 30 divisions. This scheduling problem is similar to ours. In each division a double round-robin tournament is played and the schedule is time-relaxed. All games have to be scheduled within a given period, while each team may be involved in at most one game per time slot. Home teams provide a home game set, and away teams can also specify a number of time slots in which they are not available (i.e. a forbidden game set). The teams also specify the number of time slots they want between two successive games. Unlike in our problem, the season is split in two halves, such that each team meets each other team once in each half. Furthermore, to be able to make a meaningful ranking, the season is subdivided into six time periods of equal length, and the number of games that each team has to play in each period is constrained

by a lower and an upper bound. The authors solve this problem with a permutation based genetic algorithm for which feasibility preserving operators are defined. In a follow-up paper (Schönberger et al. 2004), the authors propose a memetic algorithm, backed by a constraint propagation based heuristic, and use a co-evolutionary approach.

Knust (2010) also starts from the non-professional table-tennis scheduling problem discussed in Schönberger et al. (2000), but adds a number of constraints (e.g. some games should be played on weekend days instead of weekdays, and some games should be scheduled in specific time intervals). More importantly, for each team home and away games should be scheduled alternately (i.e. breaks should be avoided). Knust (2010) models the problem as a multi-mode resource-constrained project scheduling problem, for which an IP-formulation and a two-stage heuristic solution algorithm are proposed, involving local search and a genetic algorithm.

4 An integer programming formulation

In this section, we develop an integer programming formulation for the indoor football scheduling problem. Our main decision variable is x_{ijs}, which is 1 if team $i \in T$ plays a home game against team $j \in T\backslash\{i\}$ in time slot $s \in H_i\backslash A_j$, and 0 otherwise. The variable y_{ist} is 1 if team i plays a game in time slot s, followed by its next game in time slot t, for each $s, t \in S\backslash A_i$ such that $s < t$ and $t - s < R_{max}$, and 0 otherwise. The variable u_{ij} is 1 if no home game of team $i \in T$ against team $j \in T\backslash\{i\}$ is scheduled, and 0 otherwise. Each unscheduled game results in a non-negative penalty P. We can now write the following formulation for our problem.

minimize

$$\sum_{i\in T}\sum_{j\in T\backslash\{i\}} P u_{ij} + \sum_{i\in T}\sum_{s\in S\backslash A_i}\sum_{t=s+1}^{s+R_{max}-1} p_{(t-s+1)} y_{ist}$$

subject to

$$\sum_{s\in H_i\backslash A_j} x_{ijs} + u_{ij} = 1 \qquad \forall i, j \in T : i \neq j \qquad (1)$$

$$\sum_{j\in T\backslash\{i\}} (x_{ijs} + x_{jis}) \leqslant 1 \qquad \forall i \in T, s \in S\backslash A_i \qquad (2)$$

$$\sum_{j\in T\backslash\{i\}} (x_{ijs} + x_{jis} + x_{ijt} + x_{jit} - \sum_{k=s+1}^{t-1} (x_{ijk} + x_{jik})) - 1 \leqslant y_{ist} \qquad \forall i \in T, s, t \in S\backslash A_i : s < t, t - s < R_{max} \qquad (3)$$

$$x_{ijs} + x_{jit} \leqslant 1 \qquad \forall i, j \in T : i < j, s \in H_i\backslash A_j, t \in H_j\backslash A_i : |s - t| \leq m \qquad (4)$$

$$\sum_{j\in T\backslash\{i\}}\sum_{k=s}^{s+R_{max}-1} (x_{ijk} + x_{jik}) \leq 2 \qquad \forall i \in T, s \in S\backslash A_i \qquad (5)$$

$$x_{ijs} = 0 \qquad \forall i, j \in T, s \notin H_i \vee s \in A_j \qquad (6)$$

$$x_{ijs} \in \{0, 1\} \qquad \forall i, j \in T : i \neq j, s \in H_i\backslash A_j \qquad (7)$$

$$y_{ist} \in \{0, 1\} \qquad \forall i \in T, s, t \in S : s < t, t - s < R_{max} \qquad (8)$$

$$u_{ij} \in \{0, 1\} \qquad \forall i, j \in T : i \neq j \qquad (9)$$

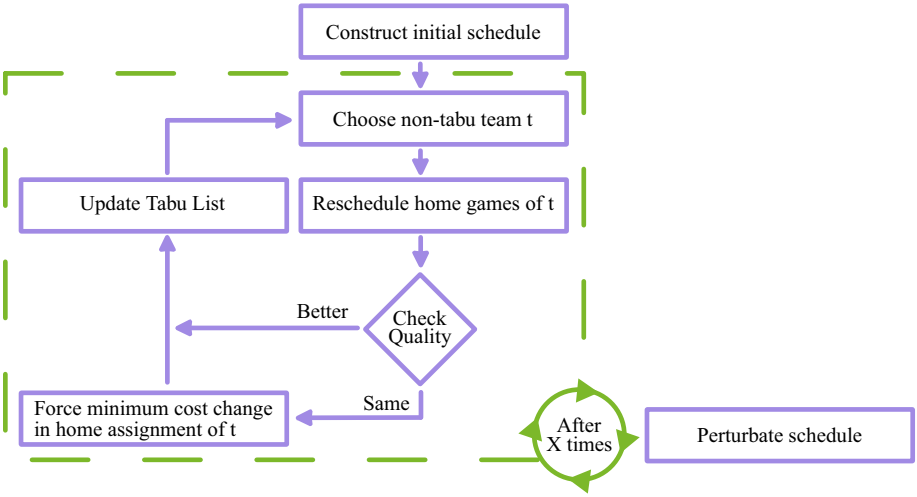

Fig. 1 General structure of the tabu search based algorithm

The objective function minimizes the number of unscheduled games, and penalizes each pair of games scheduled within R_{max} time slots. The first set of constraints ensures that each team meets each other team exactly once in a home game, unless the game is not scheduled. Consequently, if all games are scheduled, each team will meet each other team exactly once in an away game as well, and these constraints are sufficient to construct a double round-robin tournament *(C1)*. The next set of constraints makes sure that each team plays at most once per time slot *(C4)*. Constraints (3) keep track of the number of time slots between two consecutive games featuring the same team. The next set of constraints puts at least m time slots between the two confrontations of a pair of teams *(C6)*. Constraints (5) enforce that a team plays at most two games in a period of R_{max} time slots *(C5)*. Constraints (6) reduce the number of variables by explicitly stating that there is no game between two teams in a particular time slot if the home team does not have its venue available *(C2)*, or if the away team included this time slot in its forbidden game set *(C3)*. Notice that constraints (6) are in principle not necessary to obtain a feasible solution as constraints (1) already require that a game is either scheduled in a time slot respecting *(C2)* and *(C3)*, or is not scheduled at all. The final sets of constraints state that all variables are binary. Also notice that constraints (8) and (9) can be relaxed by stating that all y and u variables should be greater than or equal to zero. Indeed, the objective function, constraints (1) , and the integrality conditions on the x variables (7) are sufficient to ensure that the y and u variables are 0 or 1. Experimental results with GUROBI, however, show that this relaxation does not lead to any practical performance improvement.

5 A heuristic approach

In this section, we describe our heuristic approach, which is based on tabu search. Essentially, our heuristic consists of three phases. First, the construction phase generates an initial schedule (Sect. 5.2). Second, the tabu phase (Sect. 5.3) repeatedly selects a non-tabu team, i.e., a team that is not recently chosen. The core component of our algorithm then solves

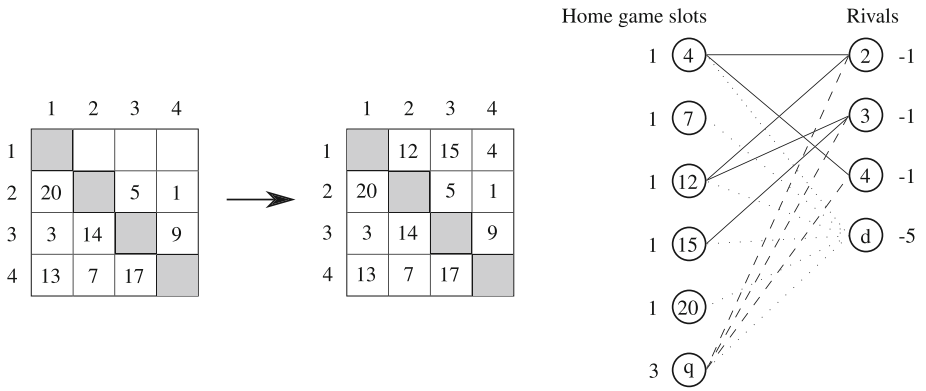

Fig. 2 Illustration of a graph G_1 with $n = 5$, $m = 5$, and $R_{max} = 3$. The leftmost matrix represents a partial schedule in which cell (i, j) corresponds with the time slot in which team i plays a home game against team j. Solving the rightmost transportation problem results in scheduling (or rescheduling) the home games of team one. Opponents are only connected with a home time slot of team one, if the assignment does not cause a conflict in the partial schedule

a transportation problem, which schedules (or reschedules) all home games of this team (Sect. 5.1). Third, the perturbation phase slightly modifies the current schedule after a certain number of iterations, which enables the heuristic to leave local optima (Sect. 5.4). Figure 1 depicts the main algorithmic structure. Finally, the last section explains how we tuned the different parameters (Sect. 5.5).

5.1 Transportation problem

Consider a partial schedule where an arbitrary number of games have been scheduled. The home game of any given team $i \in T$ can be scheduled (or rescheduled) by solving the following transportation problem (see Ahuja et al. (1993)). First, construct a bipartite graph $G_i = (U, V; E)$ as follows. We have a set of supply nodes U, containing a node with supply equal to 1 for each time slot $s \in H_i$, i.e. the home team set of team i, and a node q with supply equal to $n - 1$, corresponding to a dummy time slot. The set of demand nodes V has a node with demand equal to 1 for each opponent of team i, i.e. $T\backslash\{i\}$, and a node d corresponding to a dummy team. The demand of this last node is such that total supply equals total demand. Figure 2 represents an example of G_1.

The weights for each edge in E are set as follows. An edge from a node $u \in U$ corresponding to a home time slot $s \in H_i$ and a node $v \in V$ corresponding to a team $j \in T\backslash\{i\}$ has a cost that corresponds with inserting a home game of team i against j in time slot s in the partial schedule (solid edges in Fig. 2). This cost will depend on the previous and next game of i, and the previous and next game of j in the partial schedule, with respect to time slot s. For instance, consider the cost of the edge between time slot 12 and team two in Fig. 2. Here, the cost induced by the home team is equal to p_2 as the next away game of team one is scheduled for time slot 13. The cost induced by team two is equal to p_3 as its next game is scheduled for time slot 14. In both cases, there is no previous game within R_{max} time slots, and hence the cost of the edge is set to $p_2 + p_3$. If flow is sent from node q to some opponent j, the home game of i against j is not scheduled. Therefore, the cost of an edge from node q to a non-dummy demand node is equal to P (dashed edges in Fig. 2). Finally, the costs on

the edges between the dummy team node and any node in U are zero (dotted edges in Fig. 2). These edges correspond with unused home time slots.

Observe that the graph need not be complete for four reasons. First, if a time slot corresponding to node u is in team j's forbidden game set, then there is no edge between u and j. Second, the same edge is not present if team j already plays a game in time slot u in the partial schedule, or if j currently plays more than one game in a period of R_{max} time slots containing time slot u. Third, there is no edge between u and j if the game between j and i, at j's home venue, is planned within m time slots from u. Fourth, time slot $u \in U \backslash \{q\}$ corresponding to a home time slot $s \in H_i$ is not connected to any node $v \in V \backslash \{d\}$ if team i already plays an away game in the current schedule in time slot s, or if i already plays more than one away game in a period of R_{max} time slots containing time slot s. Solving this transportation problem will schedule (or reschedule) the home games of team i; observe that by construction the problem always has a solution.

Finally, notice that there is in fact not a full correspondence between the objective function in Sect. 4 and the costs as presented in this section. Indeed, when a team specifies two or more home time slots with less than R_{max} time slots in between, the following two problems can occur. First, when scheduling the home games of team i, we do not take into account costs related to scheduling two successive home games of i in less than R_{max} time slots (only away games of team i are considered for this). Second, and more problematic, solving the transportation problem can result in an infeasible solution as it does not forbid to play (a) more than two home games within R_{max} time slots, and (b) two or more home games in combination with one away game within R_{max} time slots. In practice, however, this has little or no effect, since teams almost never specify two home time slots with less than R_{max} time slots in between (for the majority of the teams, the home venue is available on a fixed weekday, every other week). Therefore, in the rare case that a team provides more than two home time slots within R_{max} time slots, we verify whether we have to take into account additional costs. If the generated schedule respects constraints *(C5)*, we output the adjusted objective as the final solution. Else, we solve the problem for each alternative. For instance, if a team provides two home time slots within R_{max} time slots, we solve the problem twice: once with the first time slot excluded from the home game set, and once with the second time slot. Note that this procedure will always yield a feasible solution.

5.2 Construction phase

In the construction phase, we solve the transportation problem sequentially, for each team $i \in T$. Initially, no games have been scheduled, and hence the cost on the solid edges is zero. During the construction phase, we gradually fill the schedule with games; the costs on the solid edges will increase accordingly.

We try two different initialization orders of the teams and pick the best one as a starting point. The first method repeatedly selects a team with the smallest number of available home time slots. Note that this number need not be equal to its total number of home time slots, as the team may play an away game in some home time slots in the current schedule, in which case these home time slots become unavailable. For each unscheduled home team $i \in T$, the second method computes for each competitor $j \in T \backslash i$ the number of time slots in which i can play a home game against j. The score of i is then equal to the minimum of all these numbers, and the method selects a team that has the smallest score. As an example, consider Fig. 2 where team one has two time slots to play against opponent two or three, but there is only one time slot in which it can play against team four and hence the score is one. In both

methods, ties are broken by choosing the team with the smallest team number. The end result of the construction phase is a schedule in which some games possibly remain unscheduled.

5.3 Tabu phase

Tabu search is a heuristic search procedure which goes back to Glover (1986) and which has proven its value in countless applications. The basic idea is to define a neighborhood of a solution, and next, when given a solution, an iteration consists of searching this neighborhood to identify the best solution in it. At each iteration the best solution is accepted (even if this solution is worse than the original one). To prevent cycling, certain moves are considered as *tabu* and are forbidden for a number of iterations, the tabu length (Glover 1986; Xu et al. 1998).

In our implementation, the tabu phase works with a tabu list that contains a set of teams and is initially empty. The neighborhood of a team i consists of all schedules which can be reached from the current schedule by rescheduling the home games of team i. Observe that we do not explicitly scan each individual solution in the neighborhood; instead, we find the best solution in the neighborhood by solving a transportation problem. This is the main reason why we opted for tabu search, since the transportation problem allows us to search through an exponentially large move neighborhood in polynomial time (see also Pisinger and Ropke 2010). If the resulting schedule is strictly better than the previous schedule, we accept the new schedule, add team i to the tabu list, and continue the tabu search phase with a new randomly picked non-tabu team. Note that the resulting schedule can never be worse, but may be identical to the previous schedule. Hence, if the resulting schedule has an equal cost, we impose changes to the home game assignment of team i. We do this by sequentially resolving the transportation problem, each time with a different solid edge that was part of the previous schedule removed from the graph. From these solutions we select the best one, and accept the resulting schedule. Notice that this schedule may be worse than the previous schedule but it will certainly be different. After adding team i to the tabu list, the tabu phase is again continued by randomly picking a new non-tabu team.

5.4 Perturbation phase

In order to escape local optima, the algorithm slightly changes the current schedule if no better solution was found after a specific number of iterations. To perform this change, the algorithm has access to two perturbation operators. The first operator randomly determines for each game in the schedule independently if the game is to be removed. The second operator chooses a team with a uniform probability, removes *all* the games of this team, and solves the transportation problem for this team.

5.5 Parameter tuning

Metaheuristics usually have a set of parameters that need to be set to define a search strategy. The tabu based heuristic as outlined above, involves four different tunable parameters. To begin, the tabu length controls the number of iterations during which a team cannot be selected. Note that the range of the tabu length is bounded upwards by the total number of teams minus one, as we need at least one non-tabu team in each iteration. We test tabu lengths in the set $\{1, 4, 7\}$. The second parameter regulates the number of iterations before

we check whether the algorithm is trapped into a local optimum. We try different lengths of 50, 100, 150, and 200 iterations. The last two parameters control the perturbation phase: with a probability of $\alpha\%$ the perturbation phase is performed with the first operator. Here, each game has a probability of $\beta\%$ to be removed. In the other case, which has a probability of $1 - \alpha\%$, the perturbation is performed with the second operator. We test α in the set $\{0.25, 0.50, 0.75\}$ and β in $\{0.01, 0.03, 0.05\}$. In total, we thus have 108 different parameter configurations.

During the past two decades, much effort has been made to determine appropriate parameter values based on statistical tests. One of such tests is *Friedman's two-way analysis of variance by ranks* which is at the base of F-race tuning approaches (Montero et al. 2014), and which has been successfully used to calibrate tabu algorithms in the past (Xu et al. 1998). Friedman's test is non-parametric and makes use of a block-design to compare the performance among all parameter configurations over all instances. We make use of this test in the following way. For a given set of problem instances, we first run all 108 configurations on each instance. Next, we calculate the rank of each configuration for each instance (see Montero et al. 2014; Xu et al. 1998). The null hypothesis of this test states that "all possible rankings of the candidates within each block are equally likely (Montero et al. 2014)." If Friedman's test rejects the null hypothesis, we perform pairwise tests between the best overall ranked configuration and all other configurations. If a configuration does not have a statistically significant lower quality than the best performer, it is added to the set of best configurations. From this set, we take the parameter configuration that has the largest number of configurations in the set that differ in at most one parameter value. If a tie occurs, we take the one with the best rank. This test has the benefit that (a) no assumptions are required with regard to the underlying distribution of ranks, and (b) variation in performance due to differences in input characteristics can be filtered out.

6 Computational results

We solve the indoor football scheduling problem for all major divisions for the seasons 2009–2010 till 2016–2017, which corresponds to 53 instances.[1] Divisions have between 13 and 15 teams, and the season is played from September 1st to May 31st, which results in $|S| = 273$ (or $|S| = 274$ in leap years). Generally, the home game set of a team has 4.5 time slots more than the number of opponents in the division. Notwithstanding, in three instances, a team provided less home time slots than it has opponents, which inevitably leads to at least one unscheduled game. On average, teams ask not to play a game on 14.8 time slots. In the opinion of the league organizers, it suffices to have 3 time slots between two successive games for a team (i.e. $R_{max} = 4$). Similarly, it suffices to have 60 time slots between the two confrontations of a pair of teams (i.e. $m = 60$). The penalties were chosen as follows: $p_2 = 10$, $p_3 = 3$, $p_4 = 1$. We set $P = 1000$ in order to maximize the number of scheduled games, and to be able to clearly distinguish the contribution of unscheduled games from games in close succession in the objective function value. Only in three test instances (11, 29, 45), one or two teams provided two home time slots within a period of R_{max} time slots. However, the generated schedules for all three instances respect constraints *(C5)*, and thus we only had to correct the objective values (see Sect. 5.1).

We implemented the tabu search based heuristic provided in the previous section using C++, compiled with g++ 4.8.5 using optimization flag -O3. To solve the transportation

[1] All instances and generated schedules are available from our website (www.sportscheduling.ugent.be).

Table 1 Best parameter configurations for the tabu based heuristic. The final selected parameter configuration is indicated in bold

# Iter.	α	β	Tabu	# Similar	# Iter	α	β	Tabu	# Similar
50	0.25	0.01	4	4	100	0.25	0.05	1	6
50	0.25	0.01	7	3	100	0.25	0.05	4	5
50	0.25	0.03	1	6	100	0.25	0.05	7	6
50	**0.25**	**0.03**	**4**	**7**	100	0.5	0.05	1	5
50	0.25	0.03	7	7	100	0.5	0.05	4	6
50	0.25	0.05	1	4	100	0.5	0.05	7	5
50	0.25	0.05	4	6	100	0.75	0.01	4	1
50	0.5	0.03	4	4	100	0.75	0.03	7	5
50	0.5	0.03	7	4	100	0.75	0.05	7	5
50	0.5	0.05	4	5	150	0.25	0.03	1	4
50	0.75	0.03	7	6	150	0.25	0.03	4	3
50	0.75	0.05	7	3	150	0.5	0.05	1	3
100	0.25	0.03	1	5	150	0.5	0.05	4	4
100	0.25	0.03	7	5	150	0.75	0.03	7	3

problems, we use an $O(n^3)$ implementation of Kuhn–Munkres algorithm (King 2009). The time limit was set to 30 s because, according to the league organizers, this allows to schedule all divisions (41 during the 2016–2017 season) in a very comfortable amount of time. Table 1 shows the best parameter configurations as determined by the tuning method (Sect. 5.5); Friedman's test rejected the null hypothesis with a p value of 2.13×10^{-15}. The parameter configuration with the highest number of similar configurations, and the lowest sum of ranks checks convergence after 50 iterations (column 1), applies the second perturbation operator with a probability of 25% (column 2), removes a game with the first operator with a probability of 3% (column 3), and has a tabu length of 4 (column 4). We note that, by coincidence, this most robust parameter configuration also has the lowest sum of ranks. In addition, we can draw the following conclusions from the table. First of all, it seems preferable to have a rather low number of iterations (50 or 100) before convergence is checked; this could be an indication that the heuristic scans the neighboring solution space quite efficiently. This belief is fostered by the perturbation parameters: when the number of iterations increases, it is no longer interesting to apply small changes as this would probably lead to already visited neighborhoods. Second, the use of a tabu list seems to improve performance as the minimal tabu length of 1 does not seem a very robust choice. Third, the large number of good configurations hints that the heuristic is not too sensitive for the specified parameter values.

We solve the integer programming formulation provided in Sect. 4 with GUROBI optimizer version 6.5.2, with a time limit of 14,400 s. The choice for GUROBI as a state-of-the-art solver is arbitrary; we also performed experiments with ILOG CPLEX but we did not notice any substantial performance difference. The primary concern of the LZV Cup is to support players with the practical organizations of the league at minor cost. Therefore membership fees are low, which makes the budget too constrained to buy expensive software licenses. In this regard, we also solve the IP models with COIN-OR CBC 2.9.8, a state-of-the-art free and open-source MILP solver (Lougee-Heimer 2003; Mittelmann 2016). Because COIN-OR CBC was usually not able to terminate within a reasonable amount of time, we impose a time limit of 5000 s. All integer formulations were solved in parallel mode enabled with 8 cores on a CentOS 7.3

Table 2 Results for real-life instances from seasons 2009–2010 till 2016–2017

Inst.	Teams	GUROBI		Heuristic (30s)		COIN-OR CBC		
		UB	Time	Start sol.	Best UB	Best sol.	LB	Time
1	15	2087	439	5175	2087	2087	2051	5000
2	15	80	1228	1093	81	83	60	5000
3	15	58	1670	112	58	80	13	5000
4	14	54	1137	126	54	58	40	5000
5	15	52	1687	1155	66	60	31	5000
6	15	39	799	70	39	47	23	5000
7	15	38	1173	1152	41	49	8	5000
8	15	1036	391	2099	1036	1040	1026	5000
9	15	35	2343	1110	39	49	12	5000
10	15	28	1979	118	29	35	4	5000
11	15	22	395	95	32	29	10	5000
12	14	20	13	44	20	20	20	168
13	15	16	1084	67	18	24	3	5000
14	14	1013	529	2072	1015	1013	1003	5000
15	14	13	298	1052	13	14	6	5000
16	15	8	395	1047	8	8	7	5000
17	15	7	1206	147	7	8	0	5000
18	13	6 (1)	14,400	60	6	6	0	5000
19	14	5 (2)	14,400	23	5	6	0	5000
20	15	5	263	55	5	5	3	5000
21	15	2004	926	2064	2004	2004	2000	5000
22	14	4	1595	40	6	6	0	5000
23	14	4	649	1040	4	4	0	5000
24	14	4	193	80	4	7	3	5000
25	14	4	156	36	4	4	3	5000
26	15	4 (3)	14,400	33	4	4	0	5000
27	14	3	1591	75	8	8	0	5000
28	15	3	245	69	3	6	0	5000
29	14	3	1	27	3	3	3	108
30	14	2	232	83	4	7	1	5000
31	14	1001	37	1011	1001	1001	1000	5000
32	15	1	110	25	1	1	1	3140
33	14	1	95	48	1	1	0	5000
34	15	3000	104	3040	3000	3001	3000	5000
35	14	1000	19	1010	1000	1000	1000	1289
36	14	1000	15	3011	1000	1000	1000	140
37	14	1000	4	1014	1000	1000	1000	187
38	15	0	164	31	0	0	0	432
39	15	0	67	18	0	1	0	5000
40	14	0	34	25	0	1	0	5000
41	13	0	29	19	0	0	0	149

Table 2 continued

Inst.	Teams	GUROBI		Heuristic (30s)		COIN-OR CBC		
		UB	Time	Start sol.	Best UB	Best sol.	LB	Time
42	14	0	30	14	0	0	0	307
43	14	0	10	22	0	0	0	134
44	15	0	4	24	0	0	0	236
45	14	0	3	17	0	0	0	222
46	14	0	2	8	0	0	0	189
47	14	0	2	75	0	0	0	148
48	14	0	2	6	0	0	0	120
49	13	0	2	5	0	0	0	125
50	13	0	1	0	0	0	0	121
51	13	0	1	1011	0	0	0	81
52	13	0	1	5	0	0	0	77
53	13	0	1	0	0	0	0	19

GNU/Linux based system with an Intel E5-2670 processor, running at 2.6 GHz and provided with 30 GB of RAM; the heuristic was run on the same machine but was given only a single core with 4 GB of RAM. Table 2 presents our computational results. The first two columns provide the instance number, and the number of teams in the division. The next two columns show the best upper bound found and the time needed by GUROBI to construct a schedule with this upper bound and prove optimality. Only for instances 18, 19, and 26, GUROBI was not able to terminate the search within the time limit; for these instances column 3 presents the best lower bound found within parenthesis. Although the initialization method of our heuristic is deterministic, the tabu phase is stochastic: the quality of the generated schedules may thus vary over different runs. Therefore, we use the heuristic to solve each instance 100 times (pseudo-random number generator initialized with seeds 0 to 99). Column 5 shows the objective value of the start solution; the first initialization method generated for 33 out of 53 instances a strictly better schedule than the second method. The rather high quality of the initial solutions stresses once more the effectiveness of the newly introduced move operator since these schedules were obtained after solving the transportation problem only once per team. Next, column 6 shows for each instance the best found solutions by the tabu search based algorithm over all runs. Columns 7–9 give the best found solution and lower bound found within the given computation time using the IP formulation in combination with COIN-OR CBC. High quality solutions are still obtained: for all instances the maximal number of games was planned, but in most cases an optimal solution is no longer found.

Figure 3 illustrates the variance in the performance of the tabu based algorithm over all 100 runs. Note that we deducted the non-avoidable cost of not scheduling games (e.g., for schedules of instance 1, we subtract a cost of 2000) as this allows for more compact figures and additional insights. Remarkably, for all instances and in all 100 runs, our heuristic always scheduled the maximum number of games. Further, the variance in performance turns out to add up to only a few penalty terms. What is more, the graph tends to reveal a relationship between high (modified) objective values and the difficulty of generating near-optimal schedules. Indeed, note that variance decreases and accuracy increases when the optimum lowers (the latter is also true for COIN-OR CBC; for instances 11 and 45, recall that

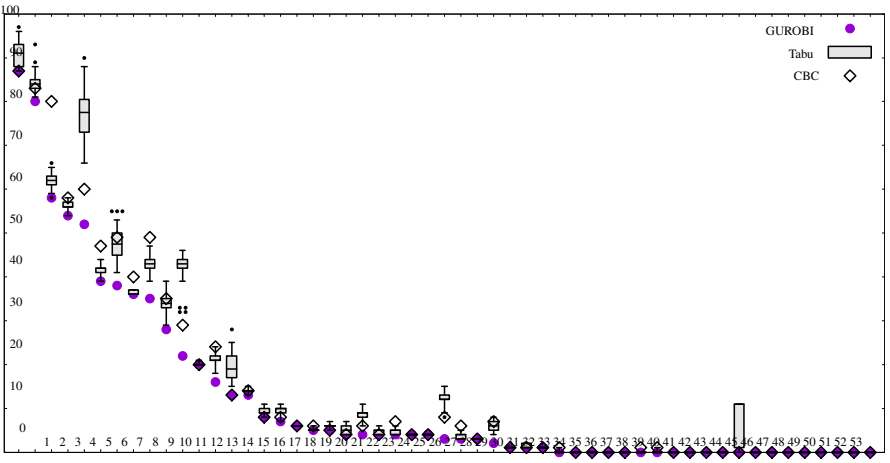

Fig. 3 Box-and-whisker plots illustrating the variance in performance over 100 independent runs. The horizontal axis represents the instance number, whereas the vertical axis represents the quality of the generated schedule. Boxes represent the three quartiles, whiskers are drawn to span 95% of the data as the use of (large) penalty terms makes the interquartile range less appropriate

one or two teams provided two home games within R_{max} time slots). Table 2 strengthens this belief as running times of both IP solvers strongly decrease whenever a low penalty solution exists. It is striking that the heuristic generates solutions of similar quality as the IP solvers, despite being single-threaded and given more than 160 times less computation time than COIN-OR CBC. Indeed, for all instances, the maximal number of games is scheduled and the objective values only differ in a few penalty terms.

7 Conclusions

In this paper, we described and solved a sports scheduling problem for the LZV Cup, a non-professional indoor football league. This scheduling problem is interesting, because games are not planned in rounds. Instead, each team has a number of time slots available to play its home games; away teams can specify time slots in which they are not able to play. Furthermore, the alternation of home and away games is irrelevant. The goal is to balance each team's games over the season, in the sense that there should be no close succession of games involving the same team. To solve this scheduling problem, we have developed a tabu search based heuristic that makes use of a novel move operator to implicitly scan the neighboring solution space by solving a transportation problem.

Prior to 2009, the association spend nearly a full week to schedule its seasons manually. Moreover, this manual approach failed to schedule a considerable amount of games and generated several congested periods. This resulted in a lot of discussion between the teams about who was responsible for the unscheduled game, and several teams complained about a close succession of games at some periods in the season. Our heuristic approach, in contrast, requires a very limited computational effort and does not require an expensive software license which makes it highly recommended for (non-professional) competitions such as the LZV Cup. The proposed heuristic has generated schedules for all seasons 2009–2010 till 2016–2017. Overall, the quality of these schedules is very close to optimal; in 42 of 53

instances the heuristic even found an optimal solution. All generated schedules were approved by the league organizers and have been implemented in practice, much to the satisfaction of the participating teams. In rare occasions where it is impossible to schedule all games, the organizers appreciate that our approach outputs a partial schedule, and that it points out which teams to contact and hold accountable.

Acknowledgements The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, FWO and the Flemish Government department EWI.

References

Ahuja, R., Magnanti, T., & Orlin, J. (1993). *Network flows: Theory, algorithms, and applications*. New York: Prentice Hall.

Alarcón, F., Durán, G., Guajardo, M., Miranda, J., Muñoz, H., Ramírez, L., et al. (2017). Operations research transforms the scheduling of Chilean soccer leagues and South American world cup qualifiers. *Interfaces*, *47*(1), 52–69.

Bao, R., & Trick, M. (2010). The relaxed traveling tournament problem. In *Proceedings of the 8th international conference on the practice and theory of automated timetabling. PATAT 2010* (pp. 167–178).

Brandão, F., & Pedroso, J. P. (2014). A complete search method for the relaxed traveling tournament problem. *EURO Journal on Computational Optimization*, *2*, 77–86.

Costa, D. (1995). An evolutionary tabu search algorithm and the NHL scheduling problem. *Information Systems and Operational Research*, *33*(3), 161–178.

Della Croce, F., Tadei, R., & Asioli, P. S. (1999). Scheduling a round-robin tennis tournament under courts and player availability constraints. *Annals of Operations Research*, *92*, 349–361.

de Werra, D. (1981). Scheduling in sports. In P. Hansen (Ed.), *Studies on graphs and discrete programming* (pp. 381–395). Amsterdam: North-Holland.

Easton, K., Nemhauser, G., & Trick, M. (2001). The traveling tournament problem description and benchmarks. In T. Walsh (Ed.), *Principles and practice of constraint programming–CP 2001* (pp. 580–584). Berlin: Springer.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, *13*(5), 533–549.

Goossens, D. R., & Spieksma, F. C. R. (2011). Soccer schedules in Europe: An overview. *Journal of Scheduling*, *15*(5), 641–651.

Goossens, D., & Spieksma, F. (2012). Scheduling the Belgian soccer league. *Interfaces*, *39*(2), 109–118.

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

King, D. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758.

Knust, S. (2010). Scheduling non-professional table-tennis leagues. *European Journal of Operational Research*, *200*(2), 358–367.

Knust, S. (2017). Sports scheduling bibliography. Resource document. http://www.inf.uos.de/knust/sportssched/sportlit_class/. Accessed 8 May 2017.

Kyngäs, J., Nurmi, K., Kyngäs, N., Lilley, G., Salter, T., & Goossens, D. (2017). Scheduling the Australian football league. *Journal of the Operational Research Society*, *68*, 1–10.

Lougee-Heimer, R. (2003). The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development*, *47*(1), 57–66.

Mittelmann, H. D. (2016). Benchmarks for optimization software. Resource document. http://plato.la.asu.edu/bench.html. Accessed 7 June 2017.

Montero, E., Riff, M. C., & Neveu, B. (2014). A beginner's guide to tuning methods. *Applied Soft Computing*, *17*, 39–51.

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

Pérez-Cáceres, & Riff, M. C. (2015). Solving scheduling tournament problems using a new version of CLONALG. *Connection Science*, *27*, 5–21.

Pisinger, D., & Ropke, S. (2010). Handbook of metaheuristics. In M. Gendreau & J. Y. Potvin (Eds.), *Large neighborhood search* (pp. 399–419). Boston: Springer.

Schönberger, J., Mattfeld, D., & Kopfer, H. (2000). Automated timetable generation for rounds of a table-tennis league. In: Zalzala, A. (Ed.), *Proceedings of the IEEE congress on evolutionary computation* (pp. 277–284).

Schönberger, J., Mattfeld, D., & Kopfer, H. (2004). Memetic algorithm timetabling for non-commercial sport leagues. *European Journal of Operational Research*, *153*(1), 102–116.

Suksompong, W. (2016). Scheduling asynchronous round-robin tournaments. *Operations Research Letters*, *44*(1), 96–100.

Westphal, S. (2014). Scheduling the German basketball league. *Interfaces*, *44*(5), 498–508.

Xu, J., Chiu, S. Y., & Glover, F. (1998). Fine-tuning a tabu search algorithm with statistical tests. *International Transactions in Operational Research*, *5*(3), 233–244.