Multi-League Sports Scheduling with Team Interdependencies: An Optimization Model

Nils Weidmann
Westdeutscher Tischtennis Verband e. V., Duisburg, Germany

Abstract

Every year, a large number of matches must be scheduled for professional and amateur sports teams. Several constraints have to be considered, including the overall capacity of venues and interdependencies between teams of the same club. As interdependent teams of a club play in different leagues, finding an optimal solution is very challenging for practitioners. While the problem of respecting capacity restrictions is well-addressed in prior work, interdependencies between teams are widely neglected, despite being a problem of major importance in practice. This paper enhances the formal definition of the multi-league-sports scheduling problem to take team interdependencies into account. We create an optimization problem to be solved by means of integer linear programming, and prove the corresponding decision problem to be $\mathcal{NP}$-complete by a polynomial reduction from 3-SAT. An implementation which was used to schedule German table tennis leagues of a certain district demonstrates the practical applicability of the approach.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Mathematics of computing → Combinatorial optimization

Keywords and phrases sports scheduling, linear optimization, constraint programming

Digital Object Identifier 10.4230/LIPIcs.CP.2025.37

Supplementary Material
Software (source code): `https://github.com/NilsWeidmann/KeyGenerator` [35]
archived at `swh:1:dir:c668e66011bff623318334bb60799385ccb4357e`
Dataset: `https://github.com/NilsWeidmann/Multi-League-Sports-Scheduling` [36]
archived at `swh:1:dir:6b32a179b07ef602689a3c799174f938fe11b379`

1 Introduction and Motivation

In several disciplines, sports teams play each other twice a season – once at home and once away – which is called a double round robin tournament (DRR). Usually, schedules for DRRs are created prior to the season, and define in which rounds (i.e., in which weeks) teams play each other. To assure fairness among teams, these schedules should be compact, such that teams play each week [4]. According to their skill, age and gender, teams are grouped into different leagues. Especially in amateur and youth sports, teams of a league are further subdivided into divisions according to regional aspects [28, 33].

Scheduling such divisions gets complicated when additional constraints have to be respected. When two teams share a venue, e.g., their home matches cannot take place in the same week. Prominent examples include AC and Inter Milan (football), New York Giants and Jets (American football) and Los Angeles Lakers and Sparks (basketball).

While creating schedules for a few professional leagues might be already challenging [1, 12], the problem gets more severe when scheduling hundreds of amateur divisions, where it is common practice that multiple teams of the same club share a common infrastructure that underlies capacity constraints. This scenario is denoted as the multi-league sports scheduling problem (MLSSP) in the following. As the number of matches which can be hosted by a club at the same time is limited, and teams that share a venue usually play in different divisions, these divisions cannot be scheduled independently.

Apart from overall capacity constraints, there are also other interdependencies between teams of a club that ensure that they play at home in complementary weeks, or in the same weeks on different days, to ensure that players can attend matches of both teams [30]. Furthermore, schedule creation is usually a distributed task: Schedulers are responsible for all divisions of a specific region allotted to them [7], which makes it necessary to respect scheduling decisions that were made on super-ordinate (e.g., national) level.

1.1 Related Work

Davari et al. propose an algorithm that minimizes the number of capacity violations of a schedule, which does not respect team interdependencies, though [7]. For the special case of all teams of a club playing at home in the same weeks without allowing capacity constraint violations, they prove this version of the problem to be $\mathcal{NP}$-complete via a polynomial reduction from vertex coloring. Li et al. extend this work by a computational study [22], a construction and improvement heuristic that minimizes constraint violations, and a multi-objective approach minimizing both traveling distances and constraint violations [23].

Other approaches rather aim at creating an optimal schedule for just a few divisions. Schönberger solves the scheduling problem on the level of single matches by means of mixed-integer programming (MIP), focusing on the optimization of player substitution opportunities [30, 31]. A computational study shows that solving the MIP involves large efforts even for small instances. Creating optimal schedules for single (professional) leagues is a well-addressed topic in prior work [2, 9, 11, 19, 24, 34]. Further research concentrates on scheduling sports competitions based on other objectives, such as balancing and minimizing breaks [3, 8, 13, 14, 18, 25, 26], traveling distances [10, 15, 16, 29], required venues [32], scheduling matches in specific rounds [5], or introducing additional matches [6, 20].

To the best of our knowledge, none of the existing approaches aims at scheduling a large number of interdependent divisions, respecting both capacity constraints, team interdependencies and constraints from super-ordinate schedules.

1.2 Contribution and Paper Organization

In this paper, we address the identified research gap by extending the definition of the MLSSP to take team interdependency constraints (which also have implications on the venue utilization) and constraints from super-ordinate schedules into account. The optimization goal is to minimize the overall number of deviations in actual and required schedules, whereby those are guaranteed to be at least similar for all teams. Thereby, the problem definition comes closer to the requirements of sports scheduling in practice. We prove the extended MLSSP to be $\mathcal{NP}$-complete by a polynomial reduction from 3-SAT, and underpin the practical applicability of our approach by scheduling German table tennis leagues of a certain district. Further experiments on generated data demonstrates the scalability of our approach.

The remainder of this paper is organized as follows: An overview of the problem domain and the proposed solution approach is provided in Sect. 2. The MLSSP is formally defined in Sect. 3, before an integer linear program (ILP) is constructed based on this definition in Sect. 4. The decision problem corresponding to MLSSP is proven to be $\mathcal{NP}$-complete in Sect. 5, and a practical implementation is sketched in Sect. 6. We apply our implementation to a real-world case study and evaluate its runtime performance in Sect. 7, before Sect. 8 summarizes the main results and sketches promising directions for future research.

2 Background and Solution Approach

Every year, schedules for thousands of (amateur) sports leagues have to be created, whereby matches between two **teams** take place at either of the teams' venues. Typically, teams of the same **club** share a common infrastructure (venues, material, etc.). The number of simultaneous matches is thus restricted by the availability of the shared infrastructure. Oftentimes, teams playing home on different days must be considered separately (e.g., those playing home on Saturdays and Sundays), turning capacity constraints into **team interdependencies**.

It is desirable to find a schedule that respects those dependency constraints to the largest possible extent. To simplify the scheduling process, matches are not scheduled individually, but on the basis of **grids** that define the order of matches throughout the season. Matches are expressed in terms of **key** pairs, whereby the keys serve as placeholders for the teams of a **division**. A concrete example illustrating the basic terminology is presented in Sect. 3.1.

The use of uniform grids for each division makes it possible to express dependency constraints involving teams playing in different divisions, as well as synchronizing schedules for leagues of different levels (e.g., national, federal state and regional level). For instance, assigning the same key to teams in different divisions ensures that they play home in **parallel**, whereas other key combinations guarantee that two teams play home in **opposite** weeks.

The overall process is depicted in Fig. 1 in form of a unified modeling language (UML) activity diagram, and comprises the following four steps:

(1) Before the schedule creation process starts, teams of a league must be grouped into divisions. **Club administrators** define interdependencies between teams of their clubs based on the availability of venues and/or tactical reasons via an internet portal. Regular match days and starting times for home matches are also defined in this step, which will come into play in step (4) when generating the match schedules.

(2) A **scheduler** assigns keys to teams in a way that dependency constraints are respected to the largest possible extent. Note that keys for clubs of which at least one team plays on a super-ordinate level are provided as additional input for the key determination process, causing a loop in the data flow.

(3) Conflict situations cannot be avoided completely in practice, as teams of neighboring clubs play against each other in multiple divisions. Thus, the scheduler resolves conflicts by assigning different, but **similar** keys to some teams, balancing interests among clubs.

(4) Finally, the scheduler enters the generated keys into the internet portal, such that match schedules can be generated based on predefined uniform grids and the regular match days and starting times as defined in step (1).

Step (2), i.e., the determination of keys for schedule creation, is a hard task to solve: The keys to be assigned to teams of the same club must be aligned, while those teams are spread over multiple divisions, and each key can only be assigned once per division. In practice, due to the lack of proper conceptual and technical solutions, this process is done manually. Schedulers try to overcome the inherent complexity by applying a divide-and-conquer approach, leading to low-quality solutions [22]. This is problematic for several reasons: First, schedulers put large efforts in finding a barely acceptable solution, whereby efforts increase disproportionately with the number of divisions to schedule [17]. Second, clubs thereupon have to eliminate the effects of bad schedules, e.g., by postponing matches or renting further venues, resulting in additional efforts on their side.

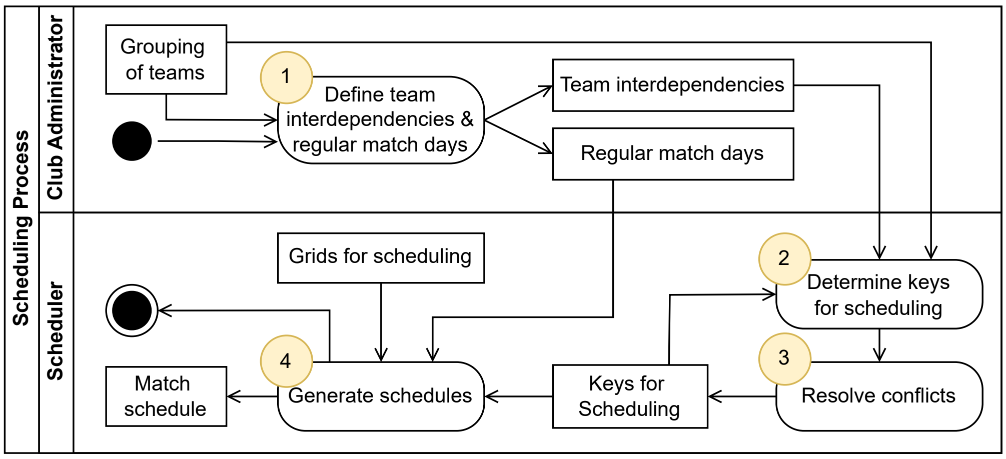

Figure 1 Process of match scheduling in double-round-robin tournaments

To overcome these hurdles, we propose to formalize the task of sports schedule creation as a combinatorial optimization problem. Instead of manually determining suitable keys, the scheduler transfers the dependency constraints of all clubs allotted to their region to a **key generator**. This software tool shall pursue the optimization goal of minimizing the number of conflict situations, i.e., situations in which the key assigned to a team differs from the required key. The advantage of modeling the problem at hand in terms of combinatorial optimization is that established SAT or ILP solvers can be integrated into the key generator, promising high performance and reliability. The final decision of how to resolve conflicts should be left to the scheduler, though, to balance the interests of the involved stakeholders. As a result, high-quality schedules can be created with substantially reduced efforts for both schedulers and club administrators.

3 Problem Definition

This section introduces the general setting and terminology formally, before team interdependencies are specified. Finally, scheduling constraints for interdependent divisions are defined, resulting in a linear optimization problem.

3.1 General Setting and Terminology

The basic concepts of the problem definition comprise clubs, teams, and divisions. Each team belongs to exactly one club and one division.

▶ **Definition 1** (Club, Team, Division). *Let $\mathcal{C}$ be a set of clubs and $\mathcal{D}$ be a set of divisions. Let $\mathcal{T}_C, C \in \mathcal{C}$ denote the set of teams of each club, and let $\mathcal{T}_D, D \in \mathcal{D}$ denote the set of teams of each division. Let $\mathcal{T} = \bigcup\limits_{C\in\mathcal{C}} \mathcal{T}_C = \bigcup\limits_{D\in\mathcal{D}} \mathcal{T}_D$ denote the set of all teams. Let $c : \mathcal{T} \to \mathcal{C}$, $d : \mathcal{T} \to \mathcal{D}$ be functions that map teams to their respective club and division.*

An example instance with 18 teams allocated to three divisions is shown in Tab. 1 (ignoring the columns “Key” and “WS” for the moment). Each team is represented by the name of its club, while most clubs have teams in multiple divisions. It is also possible that multiple teams of a club play in one division, such as two teams of “Austin” in Men’s division.

DRRs comprise multiple rounds, which are usually scheduled in consecutive weeks. To create schedules, **grids** are used that determine which matches are scheduled in which week.

Table 1 Example: Three divisions to be scheduled

Women's division			Men's division			Youth division		
Key	Team	WS	Key	Team	WS	Key	Team	WS
1	Austin	$\check{W}_1$	4	Austin	$\hat{W}_1$	1	Austin	$\check{W}_1$
2	Berkeley	$\check{W}_2$	2	Berkeley	$\check{W}_2$	3	Chicago	$\hat{W}_1$
6	Chicago	$\check{W}_1$	1	Detroit	W_0	2	Detroit	$\hat{W}_1$
5	Detroit	$\check{W}_1$	6	Green Bay	$\hat{W}_2$	5	El Paso	$\check{W}_1$
4	El Paso	$\check{W}_2$	5	Houston	W_0	6	Green Bay	$\hat{W}_2$
3	Fort Worth	W_0	3	Austin II	$\hat{W}_1$	4	Houston	W_0

▶ **Definition 2** (Grid). *Let $\mathcal{G}$ be a set of grids, with $|G| \in \{2n : n \in \mathbb{N}\}$ being the size of a grid $G \in \mathcal{G}$. Let $g : \mathcal{D} \to \mathcal{G}$ be a function that maps divisions to their grids. It holds that $\forall\ D \in \mathcal{D}\ (|\mathcal{T}_D| \leq |g(D)|)$, i.e., the number of teams in a division must be less or equal to the grid size of the division.*

An example grid for a division of up to six teams is depicted in Tab. 2. It can be read as follows: *1 - 6* in the first row of week W1 means that team 1 plays team 6 at home. The matches in W6 - W10 correspond to those in W1 - W5, swapping the home and away teams.

Table 2 Grid for divisions of a DRR with six teams

W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
1 - 6	6 - 4	2 - 6	6 - 5	3 - 6	6 - 1	4 - 6	6 - 2	5 - 6	6 - 3
2 - 5	5 - 3	3 - 1	1 - 4	4 - 2	5 - 2	3 - 5	1 - 3	4 - 1	2 - 4
3 - 4	1 - 2	4 - 5	2 - 3	5 - 1	4 - 3	2 - 1	5 - 4	3 - 2	1 - 5

The numbers in a grid serve as placeholders for the teams of a division. In the following, these numbers are denoted as **keys** (an example allocation of teams to keys can be found in the respective columns of Tab. 1):

▶ **Definition 3** (Key). *Let $\mathcal{K}_G = \{1, \ldots, |G|\}$ be a set of keys for a grid $G \in \mathcal{G}$.*

3.2 Team Interdependencies

To ensure that venues can be shared between two teams, it is important that they do not play at home in the same week. Grids have **opposite** pairs of **keys**, e.g., 1/4, 2/5 and 3/6 in the example of Tab. 2. By choosing the keys for interdependent teams accordingly, it can be assured that at most one team plays at home each week.

▶ **Definition 4** (Opposite Keys). *Let $o_G : \mathcal{K}_G \to \mathcal{K}_G, G \in \mathcal{G}$ be a total function that maps keys of a grid G to their opposite. It holds:*

$$\forall\ G \in \mathcal{G}, \forall\ K \in \mathcal{K}_G\ (o_G(o_G(K)) = K) \tag{1}$$

Using such standardized grids, team interdependencies can be defined across multiple divisions. For grids of different sizes (which are necessary for divisions with different numbers of teams), there must be a notion of which keys correspond to each other, i.e., for which keys home games are scheduled in the same weeks. Keys fulfilling this property are denoted as **parallel keys**:

▶ **Definition 5** (Parallel Keys). *Let $P_{G_1,G_2} \subseteq \mathcal{K}_{G_1} \times K_{G_2}, G_1, G_2 \in \mathcal{G}$ define a relation for parallel keys and (potentially) different grids. It holds:*

$$\begin{aligned}&\forall\ G \in \mathcal{G}, \forall\ K \in \mathcal{K}_G, ((K, K) \in P_{G,G})\\&\forall\ G_1, G_2 \in \mathcal{G}, \forall\ K_1 \in \mathcal{K}_{G_1}, \forall\ K_2 \in \mathcal{K}_{G_2}\ ((K_1, K_2) \in P_{G_1,G_2} \implies (K_2, K_1) \in P_{G_2,G_1})\end{aligned} \tag{2}$$

In contrast to professional leagues, in which matches are often shown in the media, the order of matches or the allocation of matches to certain weeks is not that critical for amateur leagues. It is far more important for clubs to define which teams should have their matches at home in opposite weeks (such that they can share one venue) or in parallel weeks (such that players can attend matches of two teams, given they do not take place on the same day).

In order to express this requirement, **week schemes** are used: Teams of a club that follow the same week scheme play at home in the same weeks, whereas teams that follow an opposite week scheme play at home in opposite weeks. There is also a distinguished week scheme W_0 to express that there are no dependencies to other teams. This can happen if a club has only one team, or the capacity of its venue(s) is sufficient to host matches of all teams at the same time (without this being an explicit requirement for the club).

▶ **Definition 6** (Week Scheme). *Let $\hat{\mathcal{W}} = \bigcup_{i=1}^{n} \{\hat{W}_i\}$, and $\check{\mathcal{W}} = \bigcup_{i=1}^{n} \{\check{W}_i\}, n \in \mathbb{N}$ be two sets of week schemes. $\hat{W}_i \in \hat{\mathcal{W}}$ and $\check{W}_i \in \check{\mathcal{W}}, 1 \leq i \leq n$ are denoted as pairs of opposite week schemes. Let $\mathcal{W} = \hat{\mathcal{W}} \cup \check{\mathcal{W}} \cup \{W_0\}$, with W_0 being a distinguished week scheme. Let $w : \mathcal{T} \to \mathcal{W}$ be a function that maps teams to week schemes. Let $\mathcal{T}_{C_W}, C \in \mathcal{C}, W \in \mathcal{W}$ be the set of teams of a club C that follow the week scheme W. It holds that*

$$\forall\ C \in \mathcal{C}(\bigcup_{W \in \mathcal{W}} \mathcal{T}_{C_W} = \mathcal{T}_C) \tag{3}$$

The "WS" column of Tab. 1 provides examples for week schemes. The women's and youth teams of Detroit follow opposite week schemes $\check{W}_1$ and $\hat{W}_1$, whereas the week scheme W_0 for the men's team indicates that there are no interdependencies to other teams. The women's and men's teams of El Paso follow independent week schemes $\check{W}_2$ and $\check{W}_1$, meaning that there are no interdependencies to each other, but potentially to other teams of the club.

Apparently, each key can only be assigned once per division (the constraint will be specified formally in Def. 12). Without weakening the requirement for opposite and parallel keys, instances of realistic size are barely feasible. A simple example can be found in the men's division of Table 1: Austin and Austin II follow the same week scheme $\hat{W}_1$, which means that they should be assigned the same key, which would directly lead to an infeasible system (which was already pointed out by Davari et al. in prior work [7]).

To overcome this non-exceptional problem, **similar keys** are defined for grids, meaning that only a few deviations from the intended sequence of home and away matches occur. In Tab. 2, one can observe that the keys 1 and 6 only differ in two weeks (W1 and W6), just as the keys 3 and 4. Therefore, keys 1/6 and 3/4, respectively, can be considered as similar keys that can be assigned in case the required key is not available.

▶ **Definition 7** (Similar Keys). *Let $S_G \subseteq {\mathcal{K}_G}^2, G \in \mathcal{G}$ define a relation for similar keys of a grid G. It holds that*

$$\begin{aligned}&\forall\ G \in \mathcal{G}, \forall\ K \in \mathcal{K}_G\ (K, K) \in S_G\\&\forall\ G \in \mathcal{G}, \forall\ K_1, K_2 \in \mathcal{K}_G\ ((K_1, K_2) \in S_G \implies (K_2, K_1) \in S_G)\end{aligned} \tag{4}$$

3.3 Key Assignment

In this subsection, we elaborate on the assignment of keys to clubs (one key per week scheme), and based on that, the assignment of keys to teams. Before keys can be assigned to week schemes of clubs, it must be clear to which grid the keys refer. Regarding teams, the grid is predefined by the divisions the teams play in, whereas such an automatism does not exist for clubs. Therefore, a **reference grid** is defined for each pair of opposite week schemes.

▶ **Definition 8** (Reference Grid). *Let $r : \hat{\mathcal{W}} \cup \check{\mathcal{W}} \rightarrow \mathcal{G}$ be a function that assigns a **reference grid** to each pair of opposite week schemes. It holds that*

$$r(\hat{W}_i) = r(\check{W}_i), i \in \{1 \dots n\}, n \in \mathbb{N} \tag{5}$$

Based on the reference grid, keys can be assigned to week schemes of clubs according to Def. 9. Note that opposite keys must be assigned for opposite week schemes, which is formalized in Eq. 6.

▶ **Definition 9** (Key Assignment for Clubs). *Let $a_W : \mathcal{C} \rightarrow \mathcal{K}_{r(W)}, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}$ be a total function that maps clubs to keys for a week scheme W. It holds that*

$$\forall\ C \in \mathcal{C}\ (a_{\hat{W}_i}(C) = o_{r(\hat{W}_i)}(a_{\check{W}_i}(C))), \hat{W}_i \in \hat{\mathcal{W}}_i, \check{W}_i \in \check{\mathcal{W}}_i, 1 \leq i \leq n, n \in \mathbb{N} \tag{6}$$

As mentioned in Sect. 1.1 and 1.2, the scheduling task is usually distributed between multiple schedulers for different levels (e.g., national, federal state and regional level). Despite having a potentially negative impact on the solution quality, this approach is indispensable to cope with the number of divisions that have to be scheduled, especially when schedules are created mostly manually. Usually, schedules for super-ordinate leagues are created first, and impose constraints for scheduling sub-ordinate leagues. In particular, key assignments for clubs that have teams in super-ordinate leagues are fixed in advance:

▶ **Definition 10** (Fixed Key Assignments for Clubs). *Let $\bar{\mathcal{C}} \subseteq \mathcal{C}$ be a set of clubs, for which there exists a fixed key assignment. Let $f_W : \bar{\mathcal{C}} \rightarrow \mathcal{K}_{r(W)}, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}$ be a total function that maps these clubs to keys for a week scheme W. It holds that*

$$\forall\ C \in \bar{\mathcal{C}}, \forall\ W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}\ (a_W(C) = f_W(C)) \tag{7}$$

The assignment of keys to teams is specified in Def. 11.

▶ **Definition 11** (Key Assignment for Teams). *Let $k : \mathcal{T} \rightarrow \mathcal{K}_{g(d(T))}$ be a total function that maps teams to keys.*

In the remainder of this section, a connection will be established between the key assignments of clubs and their respective teams. As mentioned beforehand, we must also ensure that each key is unique within a division, i.e., is assigned to at most one team per division, which is specified in Def. 12.

▶ **Definition 12** (Key Uniqueness).

$$\forall\ D \in \mathcal{D}, \forall\ T_1, T_2 \in \mathcal{T}_D\ (k(T_1) = k(T_2) \implies T_1 = T_2) \tag{8}$$

A central requirement for clubs is that – while minor deviations are allowed – teams that follow the same week scheme are assigned keys in a way that they play at home in the same weeks. Therefore, the keys of clubs and their teams must be aligned with respect to the different week schemes. In particular, each team's key must be similar to its club's key of the respective week scheme, which is specified in Def. 13.

▶ **Definition 13** (Key Similarity).

$$\begin{aligned}&\forall\ C \in \mathcal{C}, \forall\ W \in \hat{\mathcal{W}}\ \cup\ \check{\mathcal{W}}, \forall\ T \in \mathcal{T}_{C_W}, \exists\ K \in \mathcal{K}_{g(d(T))}\\ &((k(T), K) \in S_{g(d(T))} \wedge (a_W(C), K) \in P_{r(W),g(d(T))})\end{aligned} \tag{9}$$

Using the previous sets of constraints, we define **satisfiability** for problem instances of MLSSP in Def. 14.

▶ **Definition 14** (Satisfiability). *An instance m of MLSSP is satisfiable, if:*

$$\forall\ C \in \mathcal{C}, \forall\ W \in \hat{\mathcal{W}}\ \cup\ \check{\mathcal{W}}, \forall\ T \in \mathcal{T}, \exists\ a_W, \exists\ k\ ((6) \wedge (7) \wedge (8) \wedge (9)) \tag{10}$$

While it is necessary in practice to assign similar keys in case the required key is unavailable, this should be avoided whenever possible, as it can, e.g., reduce the number of capacity violations. Thus, the optimization goal is to minimize the number of teams that receive a key that is **not** parallel to the key which is assigned to their club for the week scheme the team follows. The resulting optimization problem is specified in Def. 15.

▶ **Definition 15** (MLSSP as an Optimization Problem).

$$\begin{aligned}&min.\ |\{T \in \mathcal{T} \mid w(T) \neq W_0 \wedge (a_{w(T)}(c(T)), k(T)) \notin P_{r(W),g(d(T))}\}|\\ &s.t.\ (10)\end{aligned} \tag{11}$$

4 Construction of the Integer Linear Program

In this section, the optimization problem defined in Sect. 3 is converted into a (binary) ILP that can be solved to optimality with a wide range of well-established solvers.

First, binary variables are associated with teams, representing the information whether a specific key is assigned to the team.

▶ **Definition 16** (Binary Variables for Teams). *Let $\tau_{T,K} \in \{0,1\}, T \in \mathcal{T}, K \in \mathcal{K}_{g(d(T))}$ be binary variables. Let*

$$\tau_{T,K} = \left\{ \begin{array}{ll} 1 & \text{if } k(T) = K \\ 0 & \text{otherwise} \end{array} \right\}$$

Likewise, binary variables are associated with clubs, adding the week scheme as a further dimension.

▶ **Definition 17** (Binary Variables for Clubs). *Let $\gamma_{C,W,K} \in \{0,1\}, C \in \mathcal{C}, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}, K \in \mathcal{K}_{r(W)}$ be binary variables. Let*

$$\gamma_{C,W,K} = \left\{ \begin{array}{ll} 1 & \text{if } a_W(C) = K \\ 0 & \text{otherwise} \end{array} \right\}$$

Up to here, the information whether a team received the required or a similar key was expressed implicitly. As we want to minimize the number of teams for which this is the case, further binary variables are introduced that represent this **conflict** situation. They operate as auxiliary variables and will be essential for specifying the objective function.

▶ **Definition 18** (Binary Variables for Conflicts). *Let $\xi_T \in \{0,1\}, T \in \mathcal{T}$ be binary variables.*

In the remainder of this section, the necessary constraints for a feasible solution are set up, before the objective function completes the optimization problem. We start with a set of simple but indispensable constraints that ensure that every club is assigned exactly one key per week scheme (resembling the property of a total function in Def. 9).

$$\forall\, C \in \mathcal{C}, \forall\, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}\ (\sum_{K \in \mathcal{K}_{r(W)}} \gamma_{C,W,K} = 1) \tag{12}$$

Likewise, the same must hold for each team, as k is a total function according to Def. 11.

$$\forall\, T \in \mathcal{T}\ (\sum_{K \in \mathcal{K}_{g(d(T))}} \tau_{T,K} = 1) \tag{13}$$

Additionally, it must be ensured that each key is assigned at most once per division (cf. Def. 12 for the key uniqueness requirement).

$$\forall\, D \in \mathcal{D}, \forall\, K \in \mathcal{K}_{g(D)}\ (\sum_{T \in T_D} \tau_{T,K} \leq 1) \tag{14}$$

Another set of constraints (15) ensures that opposite keys (cf. Def. 4) are assigned to opposite week schemes (cf. Def. 6).

$$\begin{aligned} &\forall\, C \in \mathcal{C}, \forall\, K \in \mathcal{K}_{r(\hat{W}_i)}\ (\gamma_{C,\hat{W}_i,K} \leq \gamma_{C,\check{W}_i,o_{r(\hat{W}_i)}(K)}), 1 \leq i \leq n \\ &\forall\, C \in \mathcal{C}, \forall\, K \in \mathcal{K}_{r(\check{W}_i)}\ (\gamma_{C,\check{W}_i,K} \leq \gamma_{C,\hat{W}_i,o_{r(\check{W}_i)}(K)}), 1 \leq i \leq n \end{aligned} \tag{15}$$

For a subset of clubs, the key assignments are already fixed by external factors (cf. Def. 10), which is expressed by the following set of constraints:

$$\forall\, C \in \bar{\mathcal{C}}, \forall\, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}\ (1 \leq \gamma_{C,W,f_W(C)}) \tag{16}$$

It remains to encode the key similarity requirement (Def. 13) into linear constraints. Two aspects have to be taken into consideration: First, the keys for teams depend on the grid used in the respective divisions, whereas the clubs' keys are determined by the reference grids per week scheme. Therefore, parallel keys have to be determined, as the grids in use are different in general. Second, the keys assigned to each of the teams must be similar (which includes equal) to one of the parallel keys determined in the previous step. As one valid key is sufficient for both steps and the previous constraints ensure that at most one key is assigned to each team and club/week scheme, all candidates in S and P, respectively, can be added up in (17).

$$\begin{aligned} &\forall\, C \in \mathcal{C}, \forall\, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}, \forall\, T \in \mathcal{T}_{C_W}, \forall\, K \in \mathcal{K}_{r(W)} \\ &(\gamma_{C,W,K} \leq \textstyle\sum_{(K,K') \in P_{r(W),g(d(T))}} (\textstyle\sum_{K'' \in S_{g(d(T))}(K')} \tau_{T,K''})) \end{aligned} \tag{17}$$

While the previous set of constraints (17) guarantees that the keys assigned to teams are similar to the keys assigned to their clubs, we need to express that it is indeed more desirable to assign parallel keys (i.e., equal keys in case of equal grids) than only similar keys. Assigning a similar (but not parallel) key can be considered as the resolution of a conflict situation, as the key is required by another team of the same division as well. Therefore, the (auxiliary) binary variables for conflicts (Def. 18) can be used to "count" the number of teams to which a similar (but not parallel) key is assigned.

$$\begin{aligned} &\forall\, C \in \mathcal{C}, \forall\, W \in \hat{\mathcal{W}} \cup \check{\mathcal{W}}, \forall\, T \in \mathcal{T}_{C_W}, \forall\, K \in \mathcal{K}_{g(d(T))} \\ &(\tau_{T,K} \leq \xi_T + \textstyle\sum_{(K,K') \in P_{g(d(T)),r(W)}} \gamma_{C,W,K'}) \end{aligned} \tag{18}$$

In accordance with Def. 15, the optimization goal can be formulated in terms of the sum of the binary variables for conflicts, which should be minimized. Adding all sets of constraints, the linear program for the MLSSP can be specified as shown in (19).

$$min. \sum_{T \in \mathcal{T}} \xi_T\ s.t.\ (12) \wedge (13) \wedge (14) \wedge (15) \wedge (16) \wedge (17) \wedge (18) \tag{19}$$

5 MLSSP is $\mathcal{NP}$-complete

The MLSSP is defined as an optimization problem in this paper, whereas the complexity class $\mathcal{NP}$ is only defined for decision problems. The proof for $\mathcal{NP}$-completeness will therefore refer to the decision problem, i.e., whether an instance of the MLSSP is satisfiable according to Def. 14.

▶ **Lemma 19.** *MLSSP* $\in \mathcal{NP}$.

Proof. To prove that MLSSP $\in \mathcal{NP}$ holds, we must show that a witness for MLSSP can be verified within polynomial runtime. As a verifier, the constraint sets defined in Eq. (12) - (18) can be used. As many constraints involve a summation or iteration over a set of keys, which depends on the size $|G|$ of the respective grid, the maximum size of any grid in use is denoted as $|G_{\max}|$ in the following. The runtime complexity of checking an inequality is $\mathcal{O}(1)$, we further assume that the same holds for applying the functions o (Def. 4) and f (Def. 10), and evaluating the relations P (Def. 5) and S (Def. 7) using suitable hash functions.

Table 3 provides an overview of the runtime complexity of verifying constraints of Eq. (12) - (18). Note that the number of similar and parallel keys is bounded by $|G_{\max}|$ for constraints (17) and (18).

Table 3 Runtime complexity to verify constraint satisfaction

Constraint	Runtime Complexity
(12)	$\mathcal{O}(\lvert\mathcal{C}\rvert \cdot \lvert\mathcal{W}\rvert \cdot \lvert G_{\max}\rvert)$
(13)	$\mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert)$
(14)	$\mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert)$
(15)	$\mathcal{O}(\lvert\mathcal{C}\rvert \cdot \lvert\mathcal{W}\rvert \cdot \lvert G_{\max}\rvert)$
(16)	$\mathcal{O}(\lvert\mathcal{C}\rvert \cdot \lvert\mathcal{W}\rvert)$
(17)	$\mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert \cdot (\lvert S_{G_{\max}}\rvert + \lvert P_{G_{\max},G_{\max}}\rvert)) = \mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert^2)$
(18)	$\mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert \cdot (\lvert P_{G_{\max},G_{\max}}\rvert)) = \mathcal{O}(\lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert^2)$
Overall	$\mathcal{O}(\lvert\mathcal{C}\rvert \cdot \lvert\mathcal{W}\rvert \cdot \lvert G_{\max}\rvert + \lvert\mathcal{T}\rvert \cdot \lvert G_{\max}\rvert^2)$

Overall, a witness for MLSSP can be verified in polynomial time, proving Lemma 19. ◀

▶ **Lemma 20.** $\forall L \in \mathcal{NP} : L \leq_p$ *MLSSP.*

Proof. We prove that MLSSP is $\mathcal{NP}$-hard by a polynomial reduction from 3-SAT, i.e., 3-SAT $\leq_p$ MLSSP. Let $\phi = (x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge \cdots \wedge (x_{n,1} \vee x_{n,2} \vee x_{n,3})$ be a formula of propositional logic in 3-CNF with n clauses $c_1, \ldots c_n$. Without loss of generality, we assume each clause to consist of exactly three literals (clauses with less than three literals can be brought into this form by doubling or tripling the literals present).

A formula ϕ' is constructed by adding a variable v – which does not occur in ϕ – to each clause of ϕ, and add a clause $(\neg v)$, such that ϕ' and ϕ are equisatisfiable:

$$\phi' = (x_{1,1} \vee x_{1,2} \vee x_{1,3} \vee v) \wedge \cdots \wedge (x_{n,1} \vee x_{n,2} \vee x_{n,3} \vee v) \wedge (\neg v) \tag{20}$$

Based on ϕ', an MLSSP instance m is constructed in the following way:

- $\mathcal{G} = \{G_2, G_4\}, |G_2| = 2, |G_4| = 4$
- $\mathcal{K}_{G_2} = \{1, 2\}, \mathcal{K}_{G_4} = \{1, 2, 3, 4\}$
- $\hat{\mathcal{W}} = \{\hat{W}_1\}, \check{\mathcal{W}} = \{\check{W}_1\}, \mathcal{W} = \{W_0, \hat{W}_1, \check{W}_1\}$, with $r(\hat{W}_1) = r(\check{W}_1) = G_2$
- $P_{G_2,G_2} = \{(1,1),(2,2)\}$, $P_{G_2,G_4} = \{(1,1),(2,3)\}$, $P_{G_4,G_2} = \{(1,1),(3,2)\}$, $P_{G_4,G_4} = \{(1,1),(2,2),(3,3),(4,4)\}$

- $o_{G_2}(1) = 2$, $o_{G_2}(2) = 1$, $o_{G_4}(1) = 3$, $o_{G_4}(2) = 4$, $o_{G_4}(3) = 1$, $o_{G_4}(4) = 2$
- $S_{G_2} = \{(1,1), (2,2)\}$, $S_{G_4} = \{(1,1), (1,2), (1,4), (2,1), (2,2), (2,3), (3,2), (3,3), (3,4), (4,1), (4,3), (4,4)\}$
- A club $C_{i,j}$ is created for each distinct $x_{i,j} \in \phi'$, and a club V is created for v
- $\bar{\mathcal{C}} = \{V\}$, $f_{\hat{W}_1}(V) = 2$
- For each clause $c_1, \ldots c_n$ in ϕ' (except for the last clause $(\neg v)$), a division D_i with four teams $T_{i,1}, T_{i,2}, T_{i,3}$ and $T_{i,V}$ is created. Each team $T_{i,j}$ corresponds to a literal $x_{i,j}$ in ϕ', $1 \leq j \leq 3$. Each team $T_{i,V}$ corresponds to the literal v.
- Teams corresponding to negative literals follow the week scheme $\check{W}_1$, those corresponding to positive literals follow week scheme $\hat{W}_1$.

▶ **Proposition 21.** *$\phi' \in$ 3-SAT $\iff$ m is satisfiable*

Proof. For each club $C_{i,j}$, let $(a_{\hat{W}_1}(C_{i,j}) = 1 \ \wedge \ a_{\check{W}_1}(C_{i,j}) = 2) \iff$ *true* is assigned to $x_{i,j}$ in ϕ', and $(a_{\hat{W}_1}(C_{i,j}) = 2 \ \wedge \ a_{\check{W}_1}(C_{i,j}) = 1) \iff$ *false* is assigned to $x_{i,j}$ in ϕ'. $\phi' \in$ 3-SAT holds if at least one literal of each clause in ϕ' yields *true*. This means that in each clause, there must be at least one positive literal with variable assignment *true* or one negative literal with variable assignment *false*.

Equivalently, to satisfy m, there must be at least one team T of a club C following week scheme $\hat{W}_1$ with $a_{\hat{W}_1}(C) = 1$ and $a_{\check{W}_1}(C) = 2$, or following week scheme $\check{W}_1$ with $a_{\hat{W}_1}(C) = 2$ and $a_{\check{W}_1}(C) = 1$ in each division D_i of m, resulting in at least one occurrence of $a_W(C) = 1$. The same must hold vice versa (at least one occurrence of $a_W(C) = 2$) for each division of m, which is trivially fulfilled because $a_{\hat{W}_1}(V) = 2$ and one team $T_{i,V}$ of club V following week scheme $\hat{W}_1$ is present in each division D_i.

This observation is depicted in Tab. 4, showing a case distinction for the eight possible cases. The columns $a_W(C_{i,j})$ contain the key assignments for the clubs $C_{i,j}$, while the columns $k(T_{i,j})$ contain the sets of possible key assignments for the teams $T_{i,j}$ according to the relations P_{G_2,G_4} and S_{G_4}. A valid key assignment for the teams $T_{i,j}$ is highlighted in bold font for the first seven cases. Only in the last case, which corresponds to all variables of positive literals being assigned *false* and all variables of negative literals being assigned *true* in the respective clause of ϕ', no valid key assignment is possible for the teams of D_i, proving Proposition 21.

Table 4 Case distinction demonstrating the equivalence of clauses and divisions

$a_W(C_{i,1})$	$a_W(C_{i,2})$	$a_W(C_{i,3})$	$a_W(V)$	$k(T_{i,1})$	$k(T_{i,2})$	$k(T_{i,3})$	$k(T_{i,V})$	valid
1	1	1	2	**1**/2/4	1/**2**/4	1/2/**4**	2/**3**/4	✓
1	1	2	2	**1**/2/4	1/**2**/4	2/**3**/4	2/3/**4**	✓
1	2	1	2	**1**/2/4	**2**/3/4	1/2/**4**	2/**3**/4	✓
1	2	2	2	**1**/2/4	**2**/3/4	2/**3**/4	2/3/**4**	✓
2	1	1	2	**2**/3/4	**1**/2/4	1/2/**4**	2/**3**/4	✓
2	1	2	2	**2**/3/4	**1**/2/4	2/**3**/4	2/3/**4**	✓
2	2	1	2	**2**/3/4	2/**3**/4	**1**/2/4	2/3/**4**	✓
2	2	2	2	2/3/4	2/3/4	2/3/4	2/3/4	✗

◀

Lemma 20 follows from Proposition 21, as $\phi \in$ 3-SAT $\iff$ $\phi' \in$ 3-SAT. ◀

▶ **Theorem 22.** *MLSSP is $\mathcal{NP}$-complete.*

Proof. Theorem 22 follows directly from Lemma 19 and Lemma 20. ◀

There are some practical implications of the proof result, which will briefly discussed in the following. The key learning is that for very large instances, some multilevel divide-and-conquer approach will still be necessary to schedule leagues from national to regional level, even with tool support. For very large problem instances (e.g., an instance involving all football leagues of Spain), the scheduling problem gets intractable from a practical point of view, even if all required data was accessible to a single person. Heuristic approaches might succeed finding a sufficiently good solution, but usually need a basic feasible solution to start with, though, which itself is challenging to determine for the MLSSP.

6 Implementation

Our approach is implemented as a C# desktop application [35]. The software architecture of our implementation is sketched in Fig. 2 in form of a UML component diagram.

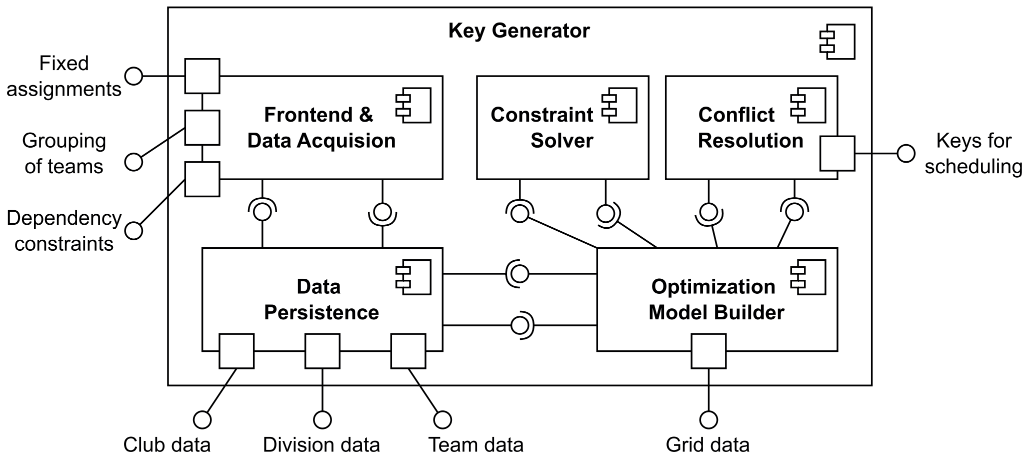

Figure 2 Component diagram for the key generator

Using the frontend & data acquisition component, the grouping of teams into divisions and the dependency constraints provided via an internet portal can be imported into the key generator via comma-separated values (CSV) files. Fixed assignments must be entered manually. The data persistence layer processes data of clubs, divisions and teams, filters out information that is irrelevant for the match scheduling task and stores the relevant information on disk. The optimization model builder constructs an ILP based on the input data and the information on parallel, opposite and similar keys for each grid in use (grids for 6 - 14 teams are supported). The ILP is subsequently solved to optimality by a SAT or ILP solver. We use the CP-SAT solver [27] in our implementation. With the conflict resolution component, the user can modify the computed solution by deciding which teams should receive similar keys in conflict situations. This makes it possible to balance interests between the clubs involved, while the solution quality remains unchanged. The data acquisition and conflict resolution components form the user interface, whereas the other components are hidden in the back end.

7 Evaluation

In this section, the practical applicability of our approach shall be investigated. Particularly, the following research questions will be answered in the course of this section:

Q1 Can the approach be used to determine solutions for scheduling instances of realistic size?

Q2 Which factors influence the scalability of the approaches, i.e., which characteristics of the input affect the runtime and solution quality of the key generation process?

Q3 How does the approach perform compared to similar existing approaches?

Outline. Q1 will be investigated by solving a real-world case study involving German table tennis leagues (Sect. 7.1). To answer Q2, the approach is applied on generated instances with different characteristic parameters (Sect. 7.2). Finally, we compare our approach to an implementation by Li et al. [22] to answer Q3 (Sect. 7.3).

Experimental setup. All experiments were run on a Windows 11 notebook with a 2.0 GHz Intel i7 processor and 32 GB main memory. The CP-SAT solver was run with 8 parallel workers (i.e., threads) and a time-out of 300 seconds. To reduce the effect of outliers, the experiments were run 10 times (Q1), or on 10 different instances that were generated with the same characteristic parameters (Q2 and Q3), respectively. From these 10 test runs, the median values of the measured runtimes and objective function values were computed.

7.1 Case Study

We applied our approach to create the schedules for a district of the West-German Table Tennis Association (WTTV) for three seasons [36]. The characteristics of the three input instances are summed up in Tab. 5. They differ with respect to the number of teams, clubs, and divisions involved. It can be further observed that for the majority of teams, dependencies were defined (i.e., week schemes other than W_0 according to Def. 6 were used). Five week schemes were available for the clubs as predefined by the WTTV. For a few clubs, key assignments were fixed by the schedulers of super-ordinate levels according to Def. 10.

Table 5 Real-world case studies from German amateur table tennis leagues

Season	# divisions	# teams (with dependencies)	# clubs (with fixed assignm.)	# week schemes	Runtime (in s)	Objective fun. value
2022/23	94	885 (638)	194 (18)	5	2.876	35
2023/24	49	516 (387)	107 (16)	5	2.145	16
2024/25	50	496 (392)	106 (11)	5	2.225	14

For all three instances, the application returns an optimal solution within a few seconds in all ten test runs. It can be concluded that the implementation is suitable for solving instances of realistic size. Limitations will be investigated in the following performance evaluation.

7.2 Performance Evaluation

As the implementation comprises an exact approach to determine the scheduling solution, the runtime is expected to increase exponentially for larger inputs. Other characteristic parameters (cf. Tab. 5) may also have an impact on the runtime and solution quality.

Base scenario construction. To investigate the approach's scalability, we derive a base scenario from the real-world case study of Sect. 7.1, which is subsequently varied with respect to its characteristic parameters. The base scenario consists of 100 *divisions* with 10 *teams per division*, which belong to 200 *clubs*. For 75% of the teams, *dependencies* are defined, and for 10% of the clubs, *key assignments* are *fixed* prior to the optimization. The number of *week schemes* is set to 5 (including W_0) in the base scenario.

Instance generation. To analyze the effect of each characteristic parameter, their values are varied separately, resulting in 30 different configurations (cf. Tab. 6). For each configuration, 10 instances were generated at random [36] and handed over to CP-SAT solver.

Table 6 Performance evaluation varying different parameters of the base scenario

Characteristic	Runtime (in s)					Objective function value				
# Divisions	20	50	100	200	500	20	50	100	200	500
	0.35	2.79	55.02	T/O	T/O	1	4	10	257	1202
# Teams per division	6	8	10	12	14	6	8	10	12	14
	0.28	1.13	2.87	2.97	3.40	2	2.5	6	8	8.5
# Clubs	20	50	100	200	500	20	50	100	200	500
	T/O	2.57	2.93	1.93	0.62	66.5	7	2.5	2.5	1.5
# Week schemes	3	5	7	9	11	3	5	7	9	11
	12.04	2.43	2.35	2.24	1.71	10	5	4	3	3
% Teams with dependencies	75	80	85	90	95	75	80	85	90	95
	2.73	2.55	2.60	2.88	3.74	5	4	4.5	5	7
% Clubs with fixed assignm.	10	20	30	40	50	10	20	30	40	50
	2.55	2.56	1.74	1.47	1.64	4.5	4	19	25.5	40

Results. The median values for runtime and solution quality are depicted in Tab. 6, while the measured values for each test run are listed in Appendices A.1 and A.2. For all conducted test runs, the CP-SAT solver returned an optimal or feasible solution, or classified the problem as infeasible. The values measured for infeasible instances were excluded from computing the median values.

The largest impact with respect to runtime and solution quality was observed when varying the number of **divisions**. For 200 and 500 divisions, the solver reached the time-out in all 10 runs, whereby the solution quality deteriorated substantially. This seems plausible, as the number of divisions corresponds to the size of the constructed ILP. Considering the number of **teams per division**, both runtime and solution quality deteriorate when increasing the division size. The effects are less pronounced than the effects of increasing the number of divisions, though, which can be explained by the increased flexibility of larger division sizes regarding key assignment.

The measured values for both runtime and solution quality show that reducing the number of **clubs** makes the problem harder. This seems valid, because an increased number of teams per club leads to more teams of a club following the same week scheme, severely restricting the possibilities of assigning suitable keys. Similarly, reducing the number of **week schemes** apparently increases the problem complexity, as more teams follow the same week scheme.

Increasing the portion of **teams with dependencies** slightly increases the runtime consumption at almost constant objective function values. The most noticeable difference occurs when increasing the portion to 95%, presumably caused by lacking flexibility regarding key assignment. Interestingly, increasing the portion of **clubs with fixed assignments** reduces the runtime but also the solution quality. An explanation could be that fixed assignments reduce the search space, but also cause unavoidable conflict situations.

In theory, one would expect the same runtime measurements and objective function values in all cells of Tab. 6 which represent the base scenario. When conducting performance tests, we varied each characteristic parameter separately, such that the base scenario occurred multiple times. As different instances were generated, the measurement results slightly differ.

7.3 Comparison to Similar Approaches

We compared our implementation to the approach of Li et al. [22], who determine optimal schedules based on capacity constraints without defining interdependencies between teams. They implemented both a heuristic and an exact approach and applied them on instances with 3,5 and 10 divisions with different numbers of teams (cf. Tab. 7). All instances were made publicly available [21], such that we were able to implement an adapter for their file format and run experiments on the same input data.

Table 7 Performance comparison to exact and heuristic approaches of Li et al. [22]

Approach	Runtime (in s) per instance type					
	3-1	3-2	5-1	5-2	10-1	10-2
Li et al. (exact)	66.33	16.55	588.10	803.71	678.28	528.63
Li et al. (heuristic)	0.02	0.11	0.79	0.01	3.97	0.93
This approach	0.16	0.29	0.05	0.06	0.09	0.10

Optimal results could be determined in all 60 test runs. The median values of the measured runtimes for 10 instances which were constructed with equal characteristics are shown in Tab. 7. Our implementation performs relatively well compared to the heuristic approach and clearly outperforms the exact approach. Interestingly, our approach takes most time for the instances with only three divisions, which can be explained by the reduced possibilities to avoid conflict situations.

The exact approach of Li et al. and our approach share the optimization goal (minimizing constraint violations) and the general setting (teams play DRR tournaments in several groups, clubs' venues have capacity constraints). The main reason for the differences in runtime performance is that the search space constructed by Li et al. is much larger, as they use decision variables for every single match, while our approach operates on the level of assigning keys to teams. Li et al. can potentially further reduce the number of constraint violations via a sophisticated allocation of bye weeks to teams, or by stretching the schedule of smaller divisions to more weeks than necessary. However, our approach allows clubs to define that two specific teams should play in turn or in parallel, while Li et al. solely aim at minimizing the number of capacity constraint violations.

7.4 Summary and Threats to Validity

Revisiting our research questions, it can be stated that our approach is suitable to be applied on real-world instances (Q1). Its scalability is mainly restricted by the number of divisions and teams involved (Q2), but performs well in comparison to existing approaches (Q3).

The conducted experiments have a few limitations to be discussed in the following. Due to the lack of real-world examples, all instances used in the experiments of the performance evaluation (Sect. 7.2) were artificially generated. The time-out was set to a rather low value, which was indeed sufficient to identify whether modifying a characteristic parameter increases or decreases runtime and solution quality. Using more than one solver for the experiments and varying the characteristic parameters in combination could give more reliable insights into performance bottlenecks.

Furthermore, the differences with respect to runtime and solution quality often varied noticeably between test runs of the same configuration (cf. Appendices A.1 and A.2). This implies that the grouping of teams into divisions has a major impact on the problem complexity, which was not covered by the performance analysis. Usually, teams are not distributed randomly over the groups, but according to regional aspects. This means that the generated instances are probably harder to solve than practical use cases.

Finally, the presented evaluation was purely technical, whereas the validity of the applied quality metrics should be evaluated in the course of a user study.

8 Conclusion and Future Work

This paper defines an extended version of the Multi-League Sports Scheduling Problem (MLSSP) that takes capacity constraints, team interdependencies and constraints from superordinate schedules into account. The MLSSP is specified in form of a linear optimization problem, the corresponding decision problem is proven to be $\mathcal{NP}$-complete. The approach was applied to scheduling German table tennis leagues of a certain district, underpinning its practical applicability. Experiments with generated instances of varying size highlight the factors influencing the approach's scalability.

In future research, meta-heuristics such as simulated annealing, tabu search or genetic algorithms should be implemented to handle even larger instances. Regarding the problem definition, the similarity requirement of assignments for clubs and teams could be replaced by a requirement for pairwise similarity of assignments for teams of the same club, leading to better results in practice. Further constraints, such as inter-club dependencies or requirements for teams playing home or away in certain weeks should be easy to include. Finally, the validity of quality measures should be assessed in an empirical evaluation with practitioners.

References

1 Fernando Alarcón, Guillermo Durán, Mario Guajardo, Jaime Miranda, Hugo Muñoz, Luis Ramírez, Mario Ramírez, Denis Sauré, Matías Siebert, Sebastian Souyris, Andrés Weintraub, Rodrigo Wolf-Yadlin, and Gonzalo Andres Zamorano. Operations Research Transforms the Scheduling of Chilean Soccer Leagues and South American World Cup Qualifiers. *Interfaces*, 47(1):52–69, 2017. doi:10.1287/INTE.2016.0861.

2 Thomas Bartsch, Andreas Drexl, and Stefan Kröger. Scheduling the professional soccer leagues of Austria and Germany. *Computers & Operations Research*, 33(7):1907–1937, 2006. doi:10.1016/J.COR.2004.09.037.

3 Dirk Briskorn. Alternative IP Models for Sport Leagues Scheduling. In Jörg Kalcsics and Stefan Nickel, editors, *Operations Research, Proceedings 2007, Selected Papers of the Annual International Conference of the German Operations Research Society (GOR)*. Springer, 2007. doi:10.1007/978-3-540-77903-2_62.

4 David Van Bulck and Dries R. Goossens. Handling fairness issues in time-relaxed tournaments with availability constraints. *Comput. Oper. Res.*, 115:104856, 2020. doi:10.1016/J.COR.2019.104856.

5 Wayne Burrows and Christopher P. Tuffley. Maximising common fixtures in a round robin tournament with two divisions. *Australas. J Comb.*, 63:153–169, 2015. URL: http://ajc.maths.uq.edu.au/pdf/63/ajc_v63_p153.pdf.

6 Mats Carlsson, Mikael Johansson, and Jeffrey Larson. Scheduling double round-robin tournaments with divisional play using constraint programming. *European Journal of Operational Research*, 259(3):1180–1190, 2017. doi:10.1016/J.EJOR.2016.11.033.

7 Morteza Davari, Dries R. Goossens, Jeroen Beliën, Roel Lambers, and Frits C. R. Spieksma. The multi-league sports scheduling problem, or how to schedule thousands of matches. *Oper. Res. Lett.*, 48(2):180–187, 2020. doi:10.1016/J.ORL.2020.02.004.

8 Dominique de Werra, Loïc Jacot-Descombes, and P. Masson. A constrained sports scheduling problem. *Discret. Appl. Math.*, 26(1):41–49, 1990. doi:10.1016/0166-218X(90)90019-9.

9 Federico Della Croce and Dario Oliveri. Scheduling the Italian football league: An ILP-based approach. *Computers & Operations Research*, 33(7):1963–1974, 2006. doi:10.1016/J.COR.2004.09.025.

10 Guillermo Alfredo Durán, Mario Guajardo, Agustina F. López, Javier Marenco, and Gonzalo Andres Zamorano. Scheduling Multiple Sports Leagues with Travel Distance Fairness: An Application to Argentinean Youth Football. *INFORMS J. Appl. Anal.*, 51(2):136–149, 2021. doi:10.1287/INTE.2020.1048.

11 Dries Goossens. Optimization in sports league scheduling: experiences from the Belgian Pro League soccer. In *International conference on operations research and enterprise systems*, pages 3–19. Springer, 2017.

12 Dries R. Goossens and Frits C. R. Spieksma. Scheduling the Belgian Soccer League. *Interfaces*, 39(2):109–118, 2009. doi:10.1287/INTE.1080.0402.

13 Dries R. Goossens and Frits C. R. Spieksma. Breaks, cuts, and patterns. *Oper. Res. Lett.*, 39(6):428–432, 2011. doi:10.1016/J.ORL.2011.09.001.

14 Mark R. Grabau. Softball Scheduling as Easy as 1-2-3 (Strikes You're Out). *Interfaces*, 42(3):310–319, 2012. doi:10.1287/INTE.1110.0559.

15 Xiaoyun Ji and John E Mitchell. Finding optimal realignments in sports leagues using a branch-and-cut-and-price approach. *Int. J. Oper. Res.*, 1(1-2):101–122, 2005.

16 Graham Kendall. Scheduling English football fixtures over holiday periods. *J. Oper. Res. Soc.*, 59(6):743–755, 2008. doi:10.1057/PALGRAVE.JORS.2602382.

17 Fabian Kleintges-Topoll. Blog: Geringe Auswirkungen durch Strukturreform. https://www.mytischtennis.de/public/mytt-bloggt/18283/, 2023. Accessed: 13 Mar 2025.

18 Sigrid Knust and Daniel Lücking. Minimizing costs in round robin tournaments with place constraints. *Computers & Operations Research*, 36(11):2937–2943, 2009. doi:10.1016/J.COR.2009.01.004.

19 Jari Kyngäs, Kimmo Nurmi, Nico Kyngäs, George Lilley, Thea Salter, and Dries Goossens. Scheduling the Australian football league. *Journal of the Operational Research Society*, 68(8):973–982, 2017. doi:10.1057/S41274-016-0145-8.

20 Jeffrey Larson and Mikael Johansson. Constructing schedules for sports leagues with divisional and round-robin tournaments. *Journal of Quantitative Analysis in Sports*, 10(2):119–129, 2014.

21 Miao Li, Morteza Davari, and Dries R. Goossens. Multi League Repository. https://www.sportscheduling.ugent.be/RobinX/multiLeagueRepo.php, 2023.

22 Miao Li, Morteza Davari, and Dries R. Goossens. Multi-league sports scheduling with different leagues sizes. *Eur. J. Oper. Res.*, 307(1):313–327, 2023. doi:10.1016/J.EJOR.2022.10.010.

23 Miao Li and Dries Goossens. Grouping and scheduling multiple sports leagues: an integrated approach. *Journal of the Operational Research Society*, pages 1–19, 2024.

24 Vardges Melkonian. An Integer Programming Approach for Scheduling a Professional Sports League. *American Journal of Computational Mathematics*, 14(4):401–423, 2024.

25 Ryuhei Miyashiro, Hideya Iwasaki, and Tomomi Matsui. Characterizing Feasible Pattern Sets with a Minimum Number of Breaks. In Edmund K. Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV, 4th International Conference, PATAT 2002*, pages 78–99. Springer, 2002. doi:10.1007/978-3-540-45157-0_5.

26 Douglas Moody, Graham Kendall, and Amotz Bar-Noy. Youth sports leagues scheduling. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, pages 283–293, 2010.

27 Laurent Perron and Frédéric Didier. CP-SAT. `https://developers.google.com/optimization/cp/cp_solver/`.

28 Diego Recalde, Ramiro Torres, and Polo Vaca. Scheduling the professional Ecuadorian football league by integer programming. *Comput. Oper. Res.*, 40(10):2478–2484, 2013. `doi:10.1016/J.COR.2012.12.017`.

29 Robert M Saltzman and Richard M Bradford. Optimal realignments of the teams in the National Football League. *European Journal of Operational Research*, 93(3):469–475, 1996.

30 Jörn Schönberger. Scheduling of sport league systems with inter-league constraints. Technical report, Diskussionsbeiträge aus dem Institut für Wirtschaft und Verkehr, 2015.

31 Jörn Schönberger. The championship timetabling problem-construction and justification of test cases. In *Proceedings of MathSport International 2017 Conference*, volume 330, 2017.

32 Ling-Huey Su, Yufang Chiu, and TC Edwin Cheng. Sports tournament scheduling to determine the required number of venues subject to the minimum timeslots under given formats. *Computers & Industrial Engineering*, 65(2):226–232, 2013. `doi:10.1016/J.CIE.2013.02.021`.

33 Túlio A. M. Toffolo, Jan Christiaens, Frits C. R. Spieksma, and Greet Vanden Berghe. The sport teams grouping problem. *Ann. Oper. Res.*, 275(1):223–243, 2019. `doi:10.1007/S10479-017-2595-Z`.

34 David Van Bulck, Dries R Goossens, and Frits CR Spieksma. Scheduling a non-professional indoor football league: a tabu search based approach. *Ann. Oper. Res.*, 275(2):715–730, 2019. `doi:10.1007/S10479-018-3013-X`.

35 Nils Weidmann. KeyGenerator. `https://github.com/NilsWeidmann/KeyGenerator`.

36 Nils Weidmann. Multi-League Sports Scheduling. `https://github.com/NilsWeidmann/Multi-League-Sports-Scheduling`, 2025.

A Appendix

A.1 Performance Evaluation Results - Runtime (in s)

Config.		1	2	3	4	5	6	7	8	9	10
# Divi-sions	20	🟩 0,365	🟩 0,232	🟩 0,301	🟩 0,435	🟩 0,246	🟩 0,359	🟩 0,483	🟩 0,335	🟩 0,208	🟩 0,512
	50	🟩 2,084	🟩 2,793	🟩 2,735	🟩 1,803	🟩 2,814	🟩 2,385	🟩 3,206	🟩 2,784	🟩 2,786	🟩 4,817
	100	🟩 51,301	🟩 54,279	🟩 58,318	🟩 112,013	🟨 302,04	🟨 302,002	🟩 55,759	🟩 50,888	🟩 44,981	🟩 49,422
	200	🟨 301,603	🟨 302,127	🟨 302,082	🟨 301,738	🟨 301,888	🟨 301,895	🟨 301,719	🟨 302,308	🟨 302,037	🟨 302,175
	500	🟨 302,417	🟨 302,403	🟨 302,439	🟨 301,462	🟨 302,024	🟨 301,689	🟨 302,085	🟨 302,635	🟨 301,829	🟨 302,617
# Teams per divis.	6	🟩 0,288	🟩 0,238	🟩 0,310	🟩 0,409	🟩 0,245	🟩 0,243	🟩 0,299	🟩 0,272	🟩 0,517	🟩 0,208
	8	🟩 1,422	🟩 1,136	🟩 1,120	🟩 0,325	🟩 0,306	🟩 0,300	🟩 1,846	🟩 1,684	🟩 0,805	🟩 1,754
	10	🟩 2,071	🟩 2,030	🟩 1,976	🟩 2,850	🟩 2,273	🟩 3,595	🟩 2,880	🟩 3,073	🟩 5,864	🟩 3,478
	12	🟩 2,525	🟩 2,528	🟩 2,332	🟩 3,060	🟩 1,503	🟩 3,002	🟩 3,045	🟩 3,320	🟩 3,451	🟩 2,936
	14	🟩 2,826	🟩 2,879	🟩 3,013	🟩 3,203	🟩 3,716	🟩 3,806	🟩 3,829	🟩 3,924	🟩 3,444	🟩 3,359
# Clubs	20	🟨 301,449	🟨 302,021	🟨 302,143	🟨 301,490	🟨 301,597	🟨 301,649	🟨 301,605	🟨 302,090	🟨 301,909	🟨 302,104
	50	🟩 2,344	🟩 2,134	🟩 2,166	🟩 2,905	🟩 2,410	🟩 2,735	🟩 4,210	🟩 2,388	🟨 301,990	🟩 5,037
	100	🟩 2,052	🟩 3,688	🟩 2,056	🟩 2,955	🟩 3,220	🟩 2,661	🟩 2,973	🟩 2,910	🟩 5,462	🟩 2,674
	200	🟩 0,439	🟩 1,794	🟩 2,855	🟩 2,629	🟩 2,060	🟩 2,629	🟩 2,474	🟩 0,524	🟩 0,509	🟩 0,529
	500	🟩 0,558	🟩 0,549	🟩 0,542	🟩 1,019	🟩 1,045	🟩 0,601	🟩 0,631	🟩 0,623	🟩 0,607	🟩 1,232
# Week schemes	3	🟨 301,637	🟩 12,035	🟧 0,372	🟩 2,616	🟩 2,683	🟩 17,639	🟨 302,044	🟨 302,012	🟩 2,625	🟩 2,576
	5	🟩 2,431	🟩 2,004	🟩 2,240	🟩 2,405	🟩 3,739	🟩 2,772	🟩 2,279	🟩 3,636	🟩 2,489	🟧 0,140
	7	🟩 3,919	🟩 1,261	🟩 2,127	🟩 3,100	🟩 0,495	🟩 4,529	🟩 3,952	🟩 0,468	🟩 0,498	🟩 2,567
	9	🟩 2,148	🟩 0,425	🟩 3,332	🟩 2,326	🟩 2,944	🟩 4,396	🟩 1,265	🟩 0,616	🟩 3,435	🟩 1,901
	11	🟩 1,457	🟩 1,164	🟩 1,756	🟩 2,245	🟩 2,501	🟩 1,765	🟩 0,550	🟩 1,671	🟩 0,512	🟩 1,803
% Teams with dep.	75	🟩 2,082	🟩 1,967	🟩 2,120	🟩 2,952	🟩 2,540	🟩 2,945	🟩 2,855	🟩 2,799	🟩 2,809	🟩 2,654
	80	🟩 2,138	🟩 2,216	🟩 2,273	🟩 2,708	🟩 3,024	🟩 2,265	🟩 2,868	🟩 2,486	🟩 2,938	🟩 2,613
	85	🟩 2,177	🟩 2,211	🟩 2,275	🟩 2,796	🟩 2,326	🟩 3,648	🟩 2,713	🟩 2,574	🟩 2,873	🟩 2,628
	90	🟩 2,315	🟩 2,214	🟩 2,487	🟩 3,026	🟩 3,292	🟩 2,746	🟩 2,793	🟩 2,995	🟩 2,964	🟩 3,065
	95	🟩 2,508	🟩 2,512	🟩 25,594	🟩 3,104	🟩 2,958	🟩 17,205	🟩 3,031	🟩 4,380	🟩 4,477	🟨 301,966
% Clubs with f. a.	10	🟩 2,032	🟩 2,398	🟩 3,187	🟩 2,473	🟩 2,439	🟩 2,437	🟩 2,726	🟩 2,647	🟩 2,628	🟩 2,835
	20	🟩 1,911	🟩 1,935	🟩 2,700	🟩 2,419	🟩 0,432	🟩 3,669	🟩 4,231	🟩 4,172	🟩 1,538	🟩 3,031
	30	🟩 2,358	🟩 1,735	🟩 0,839	🟩 2,244	🟩 0,413	🟧 0,407	🟩 0,985	🟩 2,008	🟩 1,141	🟩 3,090
	40	🟩 2,483	🟩 1,471	🟩 1,873	🟩 0,976	🟩 1,465	🟧 0,366	🟩 1,356	🟩 1,042	🟧 0,417	🟩 2,341
	50	🟧 0,303	🟧 0,301	🟧 0,322	🟩 1,443	🟩 1,365	🟧 0,324	🟩 2,944	🟧 0,361	🟩 1,832	🟧 0,308

🟩 Optimal 🟨 Feasible 🟧 Infeasible

A.2 Performance Evaluation Results - Objective Function Values

Config.		1	2	3	4	5	6	7	8	9	10
# Divi-sions	20	🟩 1	🟩 1	🟩 4	🟩 1	🟩 1	🟩 1	🟩 2	🟩 1	🟩 1	🟩 1
	50	🟩 4	🟩 2	🟩 6	🟩 5	🟩 3	🟩 5	🟩 3	🟩 6	🟩 4	🟩 2
	100	🟩 11	🟩 3	🟩 6	🟩 12	🟨 34	🟨 46	🟩 10	🟩 7	🟩 9	🟩 10
	200	🟨 262	🟨 274	🟨 212	🟨 225	🟨 246	🟨 258	🟨 275	🟨 285	🟨 256	🟨 254
	500	🟨 1202	🟨 1225	🟨 1272	🟨 1248	🟨 1202	🟨 1175	🟨 1148	🟨 1166	🟨 1218	🟨 1197
# Teams per divis.	6	🟩 1	🟩 0	🟩 2	🟩 1	🟩 4	🟩 2	🟩 2	🟩 4	🟩 1	🟩 3
	8	🟩 1	🟩 3	🟩 5	🟩 0	🟩 2	🟩 0	🟩 3	🟩 3	🟩 2	🟩 4
	10	🟩 6	🟩 3	🟩 12	🟩 7	🟩 8	🟩 3	🟩 6	🟩 6	🟩 6	🟩 5
	12	🟩 9	🟩 7	🟩 12	🟩 3	🟩 6	🟩 5	🟩 12	🟩 12	🟩 9	🟩 2
	14	🟩 3	🟩 10	🟩 11	🟩 9	🟩 3	🟩 4	🟩 11	🟩 9	🟩 8	🟩 8
# Clubs	20	🟨 68	🟨 69	🟨 62	🟨 79	🟨 68	🟨 55	🟨 60	🟨 73	🟨 65	🟨 63
	50	🟩 3	🟩 9	🟩 6	🟩 8	🟩 5	🟩 12	🟩 5	🟩 8	🟨 22	🟩 5
	100	🟩 2	🟩 6	🟩 4	🟩 0	🟩 3	🟩 2	🟩 2	🟩 1	🟩 5	🟩 4
	200	🟩 2	🟩 4	🟩 13	🟩 3	🟩 2	🟩 4	🟩 5	🟩 0	🟩 0	🟩 1
	500	🟩 0	🟩 2	🟩 1	🟩 1	🟩 4	🟩 4	🟩 3	🟩 0	🟩 1	🟩 3
# Week schemes	3	🟩 7	🟩 11	🟧 0	🟩 4	🟩 10	🟩 10	🟨 15	🟨 21	🟩 10	🟩 12
	5	🟩 2	🟩 5	🟩 8	🟩 6	🟩 4	🟩 5	🟩 5	🟩 5	🟩 9	🟧 0
	7	🟩 4	🟩 7	🟩 4	🟩 5	🟩 3	🟩 6	🟩 2	🟩 2	🟩 3	🟩 5
	9	🟩 0	🟩 1	🟩 2	🟩 3	🟩 2	🟩 6	🟩 4	🟩 3	🟩 4	🟩 3
	11	🟩 2	🟩 6	🟩 3	🟩 3	🟩 7	🟩 2	🟩 4	🟩 2	🟩 2	🟩 3
% Teams with dep.	75	🟩 11	🟩 4	🟩 6	🟩 2	🟩 6	🟩 4	🟩 6	🟩 2	🟩 3	🟩 7
	80	🟩 4	🟩 5	🟩 6	🟩 6	🟩 4	🟩 3	🟩 3	🟩 6	🟩 4	🟩 2
	85	🟩 6	🟩 5	🟩 6	🟩 3	🟩 7	🟩 5	🟩 4	🟩 3	🟩 4	🟩 4
	90	🟩 3	🟩 4	🟩 5	🟩 5	🟩 4	🟩 8	🟩 8	🟩 4	🟩 5	🟩 12
	95	🟩 8	🟩 7	🟩 6	🟩 9	🟩 9	🟩 6	🟩 4	🟩 7	🟩 5	🟨 28
% Clubs with f. a.	10	🟩 2	🟩 5	🟩 2	🟩 5	🟩 5	🟩 5	🟩 4	🟩 4	🟩 6	🟩 3
	20	🟩 4	🟩 7	🟩 4	🟩 3	🟩 3	🟩 5	🟩 11	🟩 11	🟩 2	🟩 3
	30	🟩 9	🟩 20	🟩 23	🟩 14	🟩 7	🟧 0	🟩 16	🟩 19	🟩 21	🟩 24
	40	🟩 28	🟩 38	🟩 28	🟩 20	🟩 15	🟧 0	🟩 23	🟩 20	🟧 0	🟩 31
	50	🟧 0	🟧 0	🟧 0	🟩 42	🟩 47	🟧 0	🟩 32	🟧 0	🟩 38	🟧 0

🟩 Optimal 🟨 Feasible 🟧 Infeasible