NHH

Norwegian School of Economics
Bergen, Spring 2021

Scheduling Sports Tournaments by Mixed-Integer Linear Programming and a Cluster Pattern Approach

Computational implementation using data from the International Timetabling Competition 2021

Elias Subba and Ole Jacob Lygre Stordal

Supervisor: Mario Guajardo

Master thesis, Economics and Business Administration

Major: Business Analytics

NORWEGIAN SCHOOL OF ECONOMICS

Acknowledgements

Working on this thesis has been one of the brightest points during the Covid-19 pandemic, and has helped us forget about the isolation and lack of human contact. Of course, working on this thesis has been very challenging, and at times frustrating, but the satisfaction of overcoming the problems we faced in this process made it worth every minute invested in this project.

It is with genuine enthusiasm we would like to thank our supervisor Mario Guajardo. This has been a unique occasion where we got the opportunity to work alongside a supervisor with passion, knowledge, and a deep desire to help us getting the best results possible. Without the excellent guidance and inspiring talks, we would never have been able to produce results of the same calibre.

Lastly, we would like to thank Lok Subba for read-throughs and for providing a interdisciplinary perspective.

Norwegian School of Economics

Bergen, June 2021

Ole Stordal

Ole Jacob Lygre Stordal

[illegible]

Elias Subba

Abstract

The International Timetabling Competition (ITC) has a long tradition of arranging scientific competitions within the research area of timetabling and its applications. The 2020-2021 edition is devoted to sports timetabling. The aim of ITC 2021 is to stimulate the development of solution approaches for the construction of round-robin timetables, meaning that each team plays every other team a fixed number of times. Each instance consists of a time-constrained double round-robin tournament. Further, the competition considers two types of constraints: hard constraints represent fundamental properties of the timetable that can never be violated, while soft constraints represent conditions that are desirable to satisfy. The resulting problem is to find a timetable the penalties from violated soft constraints.

In this thesis, we present a heuristic solution approach using a combination of Mixed-Integer Linear Programming (MILP) and cluster patterns for generating timetables in sports tournaments. Further, we examine how our solution approach perform on 45 experimental problem instances presented in the ITC 2021.

To the best of out knowledge, this is the first time a approach including clusters patterns, has been tested on an experimental database.

The computational results show that our solution method is capable of generating a double round robin timetable for most of the data instances. Further, it provides better results in a shorter amount of time when compared to running the default MILP model.

Keywords – Sports Timetabling, Mixed-Integer Linear Programming, International Timetabling Competition, Scheduling

Contents

List of Figures

List of Tables

1 Introduction

Sports timetabling problems can be seen as combinatorial optimization problems that consist of creating a timetable defining against whom, where, and when a team will play games. This topic has been widely researched since the 1970s (e.g. Campbell and Chen (1976)), incrementally gaining more attention ever since, and has become a specialized field with its own research conferences. Sports Timetabling problems feature a wide variety of constraints and objectives, which makes it challenging to identify the relevant set of papers for given problems. In addition, there is no generally accepted data format, which leads to problem instances and their solutions rarely being shared. In order to mitigate these issues, Van Bulck et al. (2020b) have gathered and classified different problems presented in the literature during the last five decades, in their RobinX project.

The aim of ITC 2021 is to stimulate the development of solution approaches for the construction of round-robin timetables, meaning that each team plays every other team a fixed number of times.

The thesis aims to automate the process of generating a sports timetable and the instances in the ITC 2021 are used to test our approach. Each instance contains a set of hard constraints that reflect the fundamental properties of a timetable and can therefore never be violated. Additionally, each team can have different preferences when it comes to specific games in given rounds etc., which are referred to as soft constraints in the literature. The main goal of this thesis is to create a mathematical model by using a heuristic approach consisting of a mixed-integer linear programming (MILP) model and a cluster pattern approach (referred to as CHAP).

The outline of this thesis is as follows. Section 2 provides a description of the ITC 2021. Section 3 describes the basic principles and terminology in sports timetabling. Section 4 provides a detailed description of the MILP model and solution approach. As a potential contribution to the field of sports timetabling, Section 4 also propose a heuristic approach consisting of a MILP model combined with a cluster pattern (CHAP) approach. The aim of the heuristic is to reduce the objective value, in a relatively short amount of time[1].

[1]Compared to just running the default MILP model, which can take many weeks, e.g. Klotz and Newman 2013

Utilizing team clusters can be considered a relatively new approach in sports scheduling, and has not yet been tested on experimental instances. Section 5, contains computation and implementation, where we describe the data structure, the preprocessing of data, and the implementation of our model. Section 6 summarizes our results for the heuristic approach before we present our concluding remarks in Section 7.

2 Problem Description

2.1 Principles of Scheduling

Timetable and schedule are referred to interchangeably in this thesis. In general, a schedule is considered a tool for time management and as a list of time slots where events or tasks are intended to take place. Since the 1950s, scheduling theory has become a distinct academic field within operation research, aiming to create analytical frameworks in order to optimize decision-making in real-life problems. Some classic examples of scheduling problems are the *Traveling Salesman Problem*, which concerns a series of locations to be visited while minimizing travelling distance. Another classic example considers a *nursing schedule problem*, where nurses are assigned to shifts while having to respect several constraints regarding limits on overtime and rotation of work shifts.

The ITC is a competition that aims to stimulate interest in scheduling as a field of research. The competition was first hosted in 2002 and has ever since gathered a fair share of interest in the scientific community. The type of problems featuring in ITC vary for each time (e.g., university course timetabling problems in 2019), and this year the problem consists of scheduling 2RR tournaments in sports. Table 2.1 illustrates an example of a timetable for a 2RR tournament. In the competition, the set of all teams (T) and set of all slots (P) start from 0, such that the first team and slot are Team 0 and Slot 0, respectively.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Team 0	@T1	T5	T1	@T2	T4	@T4	T2	@T3	T3	@T5
Team 1	T0	T4	@T0	@T4	T3	T2	@T3	T5	@T5	@T2
Team 2	T3	@T3	@T5	T0	T5	@T1	@T0	T4	@T4	T1
Team 3	@T2	T2	@T4	@T5	@T1	T5	T1	T0	@T0	T4
Team 4	@T5	@T1	T3	T1	@T0	T0	T5	@T2	T2	@T3
Team 5	T4	@T0	T2	T3	@T2	@T3	@T4	@T1	T1	T0

Table 2.1: 2RR schedule with 6 teams

A feasible schedule must specify in which round any two teams face each other and on whose venue. In table 2.1, the away games are denoted by @ (e.g., Team 0 plays away against Team 1 in Slot 1).

2.2 International Timetabling Competition 2021

In ITC 2021, we consider a time-constrained (compact) double round-robin tournament (2RR) with an even number of teams, where time-constrained refers to the tournament finishing using the least amount of time slots possible. 2RR describes that every team in the league will play each other twice, once at their home venue and once at the away venue, during the tournament. Thus, making the number of necessary time slots equal to $2(k-1)$ teams.

The competition releases three sets of problem instances, in which each set of instances includes 15 problems, giving a total of 45 problem instances. The number of constraints in each instance range from 93 to 1486, where each constraint $c \in C$ comes with unique subsets and parameters. In other words, every single problem instance can be represented as a large Mixed-Integer-Programming (MILP) problem. The competition considers four different sets of capacity constraints (CA1-CA4), one set of game constraints (GA1), two different sets of break constraints (BR1, BR2), one set of fairness constraints (FA2), and one set of separation constraints (SE1). A more detailed description of these constraints is given below and in the model formulation.

Timetables in sport often need to satisfy a large set of constraints. These constraints are often grouped into soft and hard constraints. The hard constraints can never be violated, such that a feasible solution can only exist if all the sets of hard constraints are respected. Soft constraints, on the other hand, represent conditions that are desirable to satisfy. A violated soft constraint will trigger a penalty with a weight specified in the XML file. When the objective is to minimize soft constraints, the general objective function, γ, states that each soft constraint violated $c \in C^S$ will trigger a penalty ($p_c = w_c \sum_{i=1}^{n_c} d_i$) that is equal to the sum of the elements of the deviation vector multiplied by the weight w_c. For all the instances in ITC 2021, the objective function sums over all the violated soft constraints.

The constraints can be grouped into the following subsets.

2.2.1 Capacity Constraints

Capacity constraints will force a team to play home or away, and regulate the number of games played by a group of teams. Four different capacity constraints are considered in the competition. The CA1 can e.g. be interpreted as preferences of playing at least a given number of home games in the most lucrative time slots, in order to increase ticket revenues. CA2 can be seen as a generalization of CA1, where one could model top and bottom teams, to avoid that bottom teams play all initial games against top teams. The ITC 2021 considers at most two CA3 hard constraints per instance, which are used to limit the maximal length of home stands (and/or away trips) by forbidding consecutive home breaks (and/or consecutive away breaks). Moreover, there can be many soft constraints that limit the total number of consecutive games against certain strength groups.

While CA2 and CA3 define restrictions for each team in different subsets of T (set of all teams), CA4 considers these subsets as single entities. CA4 is usually used to limit the total number of games between top teams over the lifespan of the tournament, known as *global* mode in the ITC 2021. It can also be used to limit the number of games per time slot, referred to as *every* mode. The latter could be the case of two teams sharing the same stadium, where you simply limit the number of home games for the respective teams in each round. Note that the capacity constraints are limit the number of games, while if you want to regulate specific games in a given time slot, then the game constraints are to be used.

2.2.2 Game Constraints

Game constraint, GA1, deals with fixed or forbidden games. For a given time slot a given match could be forbidden, or fixed. For instance, broadcasters often want to have a “top match” or a “classical match” in a given time slot, especially towards the end of the season, where they would try to fix a match between title contenders. On the other hand, police will often forbid a “high risk” match to occur in a given time slot (Van Bulck et al., 2020b). The latter could be the case if you have several teams in a city, and you would then try to avoid two teams playing the same day due to the risk of clashes between rival hooligans.

2.2.3 Break constraints

Break Constraints regulate when breaks occur, as well as the frequency. A team has a break when it has the same home-away status as in the previous game. The break constraints are either used to limit the number of breaks over a given set of rounds for given teams or to limit the total number of breaks in the competition. Usually, one would like to avoid breaks as they can affect the game attendance (Forrest and Simmons, 2006), and can be perceived as unfair due to the home-away effect (Pollard and Pollard, 2005) BR1 can forbid breaks at the beginning (end) of the season or limit the total amount of breaks per team. BR2, on the other hand, can limit the total number of breaks in a tournament.

2.2.4 Fairness Constraints

Fairness constraints (FA2) are used to increase fairness in a competition. An unequal amount of home games a team has played before a certain match can give an advantage for one of the teams. These constraints can be used to balance the number of home matches so they have a relatively equal amount of home matches before a given match. The allowed difference of home matches played, to a certain point, is specified in the constraints.

2.2.5 Separation Constraints

In order to regulate the symmetry and the number of time slots between consecutive meetings involving the same teams, one would use *separation constraints*. In a 2RR tournament, every encounter is played two times, with the difference being the home-away status of the teams (Van Bulck et al., 2020a). E.g.if Arsenal was to meet Chelsea at home in the first round of EPL[2], one would make sure that they do not meet again in the immediate future. Through the SE1 constraint, organizers may request that two matches containing the same opponents are separated by at least a given number of time slots.

[2]the English Premier League

2.3 Research Questions

In light of the ITC 2021, we attempt to formulate MILP models for all the 45 problem instances. Furthermore, we would like to introduce a cluster pattern approach with the purpose of finding good solutions within an acceptable time limit.

The following research questions are formulated:

1. How can we generate a timetable for a sports competition when many- and perhaps conflicting conditions are present?

2. How will a heuristic cluster pattern approach perform on the experimental instances?

3 Background

In this section, we elaborate on the background of sports timetabling, where we provide a literature review and methodological aspects that concerns how similar problems are modeled. Lastly, we also mention some of the practical aspects of sports timetabling.

3.1 Sports Scheduling

Over the last decades, the interest in sports, in general, has increased rapidly. The large tournaments are followed by millions of fans all over the world, which are eager to get their hands on the latest news regarding their favorite teams. This has lead to sports evolving into a large industry, with the largest tournaments playing a big part in the increasingly globalizing economy. These big tournaments bring thousands of jobs, urban regeneration, and economic opportunities to their hosts (Kendall et al., 2010).

Football, in particular, has become one of the largest subsets of the sporting industry, and with the large size comes many stakeholders. There are millions of fans that demand fairness in the competition. In order to stay competitive in the league, the teams are willing to invest millions of dollars into players, stadiums, advertising, merchandising and broadcasting rights. Furthermore, the organizers, airlines, police, media , and players also play a vital part in the tournaments and want to have their say in the scheduling of the tournaments. As a result of this, the interest in sports scheduling have increased intact with the size of the sport. Over the last decades, academic papers on the topic of sports timetabling has increased in numbers of publications and evolved into a large field of research of its own (Goossens, 2018).

3.2 Terminology

In order to make it easier to understand the content of the thesis, we introduce the basic terminology in sports timetabling. The RobinX paper (Van Bulck et al., 2020b) summarises the general terminology that is used within the field of sports scheduling problems. Below we elaborate on the concepts relevant for understanding the nature of the ITC 2021 problem instances.

The problem instances consist of a set of slots S, a set of teams T, and an ordered multiset of games G. Every $k \in S$ represents a time period or a game day, where every team can play at most one game. If the team does not play a game in a time slot, we say that the team has a *bye*. The multiset G consists of ordered pairs (i, j), where every $i \in T$ represents the home team, and $j \in T$ represents the away team. The variable $x_{i,j,k}$ represents which home team i, hosts the venue for the away team j, in the time slot k.

Breaks are a fundamental element of sports timetabling. We say that a team has a break if they play two home games in a row or if a team plays two away games in a row. In other words, a break occurs if a team plays two consecutive games with the same *home-away status*. In this competition, a break takes place in the time slot of the second occurrence (E.g., Team1 plays a home game in slot 1, home-game slot 2, a break occurs in slot 2.).

In essence, we aim to create schedules that maximize the demands of the different stakeholders. Goossens (2018) explains that a schedule is a combination of a *Home- and Away Pattern* (HAP) and a timetable. The HAP is a sequence that determines if a team plays home or away in a certain round. For example, if Team 1 has a HAP = (0,1,0,1), it means that Team 1 plays away-home-away-home. The timetable, on the other hand, determines which teams are set to play each other in each round. Ultimately, these two components make up a complete timetable. It is important to note that the schedule needs to be created by complementary patterns. For every time slot in the tournament, patterns must be compatible, meaning that for every team that plays a home game, an opponent must play an away game.

3.3 Round Robin Tournaments

The problem instances are using the format Double Round Robin, hereby referred to as *2RR*. In a 2RR tournament, every team plays each other two times over the tournament, while the home/away status is alternating between the two games. According to Kendall et al. (2010), 2RR tournaments are one of the most researched tournament formats, as the format is common practice for many of the biggest sports tournaments in the world.

3.3.1 Competition Format

As previously mentioned the RobinX paper (Van Bulck et al., 2020b) created a uniform three-field system to describe the structure of sporting tournaments.

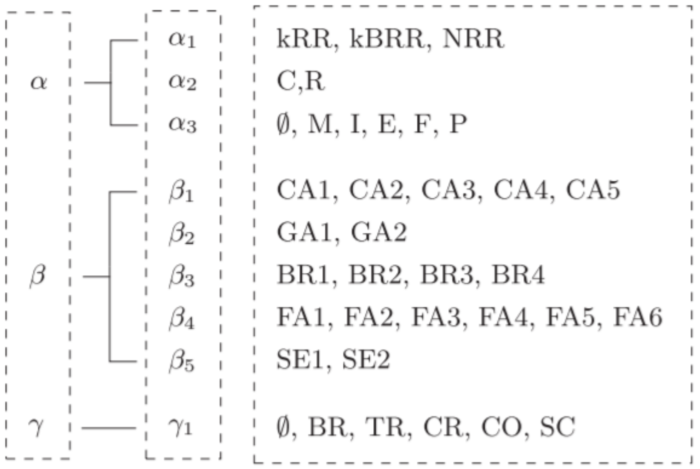
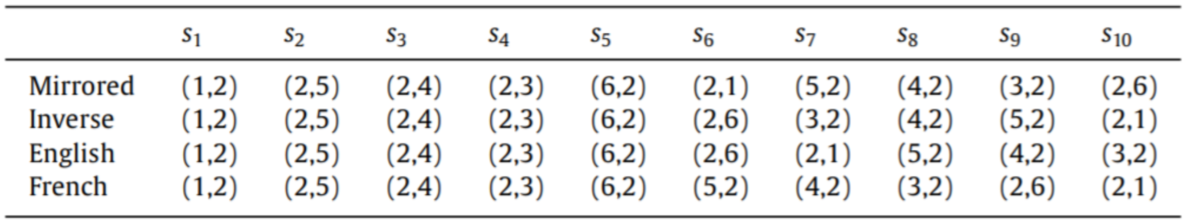


Figure 3.1: Overview of the three-field notation for sports timetabling. (Van Bulck et al., 2020b)

Starting with the α field, all the problem instances are using $\alpha_1 = 2RR$ and $\alpha_2 = C$. This means that the entirety of our tests is performed on *compact* 2RR tournaments. A *compact* tournament means that the number of rounds in a tournament is restricted to the bare minimum, which equals $n(2-1)$, where n represents the number of teams. In some of the instances, we deal with problems where $\alpha_3 = P$. This means that the game mode is *phased*. In a *phased* tournament, the competition consists of two consecutive 1RR's so that the teams are required to meet one time in each half of the season.

When scheduling a 2RR tournament, it is common to structure the competition using a symmetric scheme (Goossens and Spieksma (2012)). The most common symmetric schemes are *Mirrored*-, *Inverted*-, *English*- and *French* scheme, which are expressed in the α_3 field as M,I,E,F. Though our problems are not originally subjected to any of these symmetric schemes, we find it appropriate to present the ideas behind them as we perform some experiments based on these schemes.

In a *mirrored* tournament, we start by creating a solution for the first half of the season. Then, the second half is created as close to identical to the first half, with the only difference being the home-away status of the teams is reversed. The *Inverted* scheme is based upon the same principle, where they begin to create the first half of the season. Then, the second half of the season is created, but unlike the mirrored format, the meetings in the second half are in reversed order of the meetings in the first half. The *English-* and *French scheme* have also been used in many European competitions (Goossens and Spieksma (2012)). Tournaments that are scheduled using the *French scheme*, the games in the first and last time slots are equal along with the games in slot $n-1+k$ and slot $t+1$ where $k=1,2...,n-2$, with the team playing at home being inverted. Finally, in the *English scheme*, meetings in the last time slot of the first half are equal to the first meeting in the second half. The slot $n+k$ in the second half also corresponds to slot k in the first half for slots $k=1,2...,n-2$. These 4 schemes were also used by Durán and M. Guajardo (2017) when scheduling the South American qualifiers of the FIFA world cup 2018.



	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}
Mirrored	(1,2)	(2,5)	(2,4)	(2,3)	(6,2)	(2,1)	(5,2)	(4,2)	(3,2)	(2,6)
Inverse	(1,2)	(2,5)	(2,4)	(2,3)	(6,2)	(2,6)	(3,2)	(4,2)	(5,2)	(2,1)
English	(1,2)	(2,5)	(2,4)	(2,3)	(6,2)	(2,6)	(2,1)	(5,2)	(4,2)	(3,2)
French	(1,2)	(2,5)	(2,4)	(2,3)	(6,2)	(5,2)	(4,2)	(3,2)	(2,6)	(2,1)

Figure 3.2: 2RR League with 6 teams, illustrating games of Team 2 (Van Bulck et al., 2020b)

The purpose of the symmetric schemes is to implement fairness into the competition, as leagues that are using symmetry have a considerable amount of games between mutual meetings.

The β-field lists the constraints and are divided into five classes, which were elaborated in the previous chapter. Lastly, the γ-field refers to the objective function in use. As the competition only considers problems revolving around the minimization of soft constraints, we do not elaborate further on the other problem formats.

3.4 Techniques Used for Sports Scheduling Problems

Kendall et al. (2010) summarise over 160 journals on the topic of sports scheduling up until 2010, where they review different literature and modeling techniques. They argue that when it comes to solving real-life problems, the choice of methodology is crucial in order to obtain acceptable results. The authors further claim that one of the most common practices is to develop an integer program (IP) that aims to maximize or minimize an objective function. IP models can be very effective at solving sports scheduling problems, and in a round-robin tournament they found that most problems use the following variable definition:

$$x_{i,j,k} \begin{cases} 1 & \text{if team i plays against team j in round k} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

With an n number of teams, where the teams $i, j = 1...n$, and time slots $k = 1...s$. A 2RR tournament can be formulated using the following constraints:

$$\sum_{k=1}^{s}(x_{i,j,k}) = 1 \qquad \forall i, j \quad i \neq j \tag{3.2}$$

$$\sum_{j=1}^{n}(x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall i, k \tag{3.3}$$

These constraints set the base for the vast majority of IP models. The first constraint forces each team to play against every other team once at home during the tournament, while the second constraint subjects all teams to only being able to play one game every round.

3.4.1 Two-Phased Approach

Goossens (2018) argues that as new effective algorithms and hybrid models (e.g., Rasmussen 2006) have emerged over the last decade, so has the size of the problems. The debate for more solution approaches is open, and one of the most important additions to the field of research is the two-phased approach, also known as *the decomposing method*, introduced by Nemhauser et al. (1998). This approach is based around the principle of *first break, then schedule*, which in the first phase assigns a team to a HAP. Subsequently, the second phase consists of deciding the opponents for each team in each time slot, subject to their HAP. By subsequently solving the two phases, an optimal solution is not guaranteed, and typically there should be room for improvement.

The first phase can be modeled by the following formulation:

$$h_{i,p} \begin{cases} 1 & \text{if team i is assigned to pattern p} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$\sum_{p} h_{i,p} = 1 \qquad \forall i \tag{3.5}$$

$$\sum_{i} h_{i,p} = 1 \qquad \forall p \tag{3.6}$$

The last two constraints force each team to be assigned to home and away patterns. After the solver has assigned each team to a pattern, equations (3.4), (3.5), and (3.6) can be used to model the second phase, with the exception that $x_{i,j,k}$ takes the value allowed by the HAPs of team i and j.

The formulations above only represent the main ideas behind the two phases. In reality, there exist many different methods on how the patterns can be generated, and how the phases can be structured and solved. E.g., Nemhauser et al. (1998) used integer programming to solve phase I and II but added a third enumeration phase when scheduling the major college basketball conference.

3.4.2 Literature Review

There is no secret that solving large MILP models can be a daunting task. When we started working on our thesis, we ran the original model on 10 instances and were only able to produce one feasible solution. This is a common case in sports scheduling, and Durán et al. (*forthcoming*) also claim that some of their instances could run for days without even finding a feasible solution. When the search parameters are too wide to produce acceptable results in a reasonable time, variations of the two-phased approach have been used to address these problems, as it greatly reduced the size of the MILP models (Kendall et al., 2010).

The decomposing method by (Nemhauser et al., 1998), where MILP models are combined with a heuristic search process through a large pool of predefined HAPs, has laid the foundation for scheduling in many important football tournaments. Hausken et al. (2013) scheduled the Norwegian football league while building on the same principles, where they created an algorithm that aimed to generate complimentary HAPs with a low number of breaks. They further created a MILP model that would match each team to a pattern while minimizing the total amount of breaks in the tournament. Bartsch et al. (2006) scheduled both the *German Bundesliga* and the *Austrian Football Bundesliga*, while Recalde et al. (2013) scheduled the *Ecuadorian Serie A*, all based on the same principles of an iterative search process of predefined H-A patterns. Similar techniques has also been used to professional football leagues in Chile (Durán et al. 2007, Durán et al. 2012, Alarcón et al. 2017), Ecuador (Recalde D, 2013), Denmark (Rasmussen, 2008), Honduras (Fiallos et al., 2010).

When (Goossens and Spieksma, 2009) elaborate on their experience from scheduling the 07-08 season in the Belgium football league, they mention some additional advantages the decomposition method brings when working on real-life problems. As opposed to theoretical problems, in real-life problems, the participating teams often want to suggest minor adjustments in the schedule. The authors further argue that the "*first-break-then-schedule*" approach is robust to changes, allowing teams to make some minor adjustments without having the rebuild the entire schedule from scratch. Further, the constraints regarding breaks often originate from the police or stadium unavailability and are not likely to change, meaning that once a feasible set of HAPs has been generated, the process does not have to be repeated.

In the forthcoming paper of Durán et al. (2021), they describe their work when scheduling Argentina's professional football league, *Superliga*. They explain that they formed an IP model and used a *decomposition* approach, but unlike any other previous work, it was based on the assignment of *cluster patterns* (CHAP) instead of HAPs. As explained earlier, the teams' HAPs are vectors that explain the home-away status for every game and every team in the tournament. The CHAPs, on the other hand, expands on the methodology and can be described as HAPs only for subsets of the teams. By using this approach, they were able to find high quality solutions within a few seconds or minutes, even solving to optimality in some instances. Ultimately they concluded that a heuristic CHAP approach could improve the results in more general sports scheduling problems.

We believe that the ITC2021 competition is a perfect opportunity to experiment on how a CHAP approach could be applied in more general sports scheduling problems.

4 Model Formulation and Solution Approach

In this section, the instances are modeled as an optimization problem using the mixed-integer linear programming (MILP) approach. First, we present our MILP model. Second, we implement a heuristic approach based on MILP and a cluster pattern approach, referred to as CHAP.

In general, our approach is based on MILP modeling followed by some adjustments as we use different strategies within this approach to run the modeling in exact or in approximated form, depending on if we are only searching for feasibility or optimality. We adjust the model to run in different ways, which are more or less constrained and with slightly different objective functions.

4.1 Model Formulation

Sets

T : Set of all teams

T_c^1 : First indexed subset of teams for every constraint

T_c^2 : Second indexed subset of teams for every constraint

P : Set of all rounds

S_c : Indexed Subset rounds for every constraint

G_c : Indexed multiset of ordered pairs (i,j) for every constraint

C : Set of all Constraints

The logic behind the definition of the subsets of all constraints, C, is as follows. If we use $CA2^{HHA}$ as an example, the first letter indicates whether the respective constraint is a hard or soft constraint, followed by mode HA, H or A. So, $CA2^{HHA}$ concerns the CA2 hard, home and away constraints.

Variables

We use the most common approach (Kendall et al., 2010) and introduce the family of variables x that define which teams play against each other, in round k, where i represents the home team and j the away team, if a feasible solution is found. For the purpose of handling the break constraints, we introduce the variables h and a, which determines in which time slots k, team i have a home or away break. In addition, we introduction variables d that counts the number of deviations for the soft constraints. A detailed model is locate in the appendix.

$$x_{i,j,k} \begin{cases} 1 & \text{if team i plays against team j in round k} \\ 0 & \text{otherwise} \end{cases}$$

$$h_{i,k} \begin{cases} 1 & \text{if team i has a home break in round k} \\ 0 & \text{otherwise} \end{cases}$$

$$a_{i,k} \begin{cases} 1 & \text{if team i has an away break round k} \\ 0 & \text{otherwise} \end{cases}$$

Constraints

The constraints (4.1),(4.2),(4.3),(4.4), and (4.5) represent the base of our models and are present in all of the 45 problem instances. Constraint (4.6) and (4.7) are present in the problems containing a *phased* game mode.

$$x_{i,i,k} = 0 \qquad \forall i \in T, k \in P \tag{4.1}$$

$$\sum_{j \in T} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall i \in T, k \in P \tag{4.2}$$

$$\sum_{k \in P} x_{i,j,k} = 1 \qquad \forall i \in T, j \in T : i \neq j \tag{4.3}$$

$$\sum_{j \in T} (x_{i,j,k_{-1}} + x_{i,j,k}) - b_{i,k} \leq 1 \qquad \forall i \in T, k \in P : k > 0 \tag{4.4}$$

$$\sum_{j \in T} (x_{j,i,k_{-1}} + x_{j,i,k}) - a_{i,k} \leq 1 \qquad i \in T, k \in P : k > 0 \tag{4.5}$$

$$\sum_{k=1}^{|P|/2} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall k \in P, i \in T, j \in T : i \neq j \tag{4.6}$$

$$\sum_{k=(|P|/2)+1}^{|P|} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall k \in P, i \in T, j \in T : i \neq j \tag{4.7}$$

$$x_{i,j,k} \in \{0,1\} \qquad \forall i,j \in T, k \in P \tag{4.8}$$

$$h_{i,k} \in \{0,1\} \qquad \forall i \in T, k \in P \tag{4.9}$$

$$a_{i,k} \in \{0,1\} \qquad \forall i \in T, k \in P \tag{4.10}$$

Constraints (4.1-4.3) make up the base for a 2RR season (Kendall et al., 2010). Constraint (4.1) is included for ease of notation and implementation of the code. Constraint (4.2) forces each team to play exactly one game per round, while constraint (4.3) ensures that each team meets during a season. Some of the instances are *phased*, meaning that the overall structure of the league is constructed using two consecutive 1RR‘s. Constraint (4.6) and (4.7) force each pair of teams to play one time in the first half of the season and one time in the second half of the season. In ITC 2021 breaks are defined as playing a game with the same home-away status as in the previous game. E.g., if a team is playing home in Slot 2, as well as playing home in Slot 3, the team enters a break in Slot 3. Constraint (4.4) and (4.5), ensure that the correct values are assigned to the home and away break variables. Constraints (4.8-4.10) define the variables as binary variables.

Capacity Constraints

As mentioned in the problem description, capacity constraints aim to control the number of home and away games played by a team and regulates the total number of games played by a team or group of teams.

The set of CA1 put restrictions on the number of games a given subset of teams can play in a given subset of slots. E.g., a team from T_c^1 can only play max_c games in the specified subset of slots S_c. Constraints (4.11) and (4.12) limit the number of home(H) or away(A) games for each team i in T_c^1 during a subset of slots defined in S_c for all hard constraints.

$$\sum_{k \in S_c} \sum_{j \in T} x_{i,j,k} \leq max_c \qquad \forall c \in CA1^{HH}, i \in T^1 : i \neq j \tag{4.11}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{j,i,k} \leq max_c \qquad \forall c \in CA1^{HA}, i \in T^1 : i \neq j \tag{4.12}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{i,j,k} - max_c \leq d_{i,c}^{CA1} \qquad \forall c \in CA1^{SH}, i \in T^1 : i \neq j \tag{4.13}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{j,i,k} - max_c \leq d_{i,c}^{CA1} \qquad \forall c \in CA1^{SA}, i \in T^1 : i \neq j \tag{4.14}$$

(4.13) and (4.14) defines the soft version of the constraints, in which the number of home and away games exceeding max_c for each constraint, is counted.

The set of CA2 constraints can be seen as a generalization of CA1, where the constraints also specify the opponent team. E.g. Team i from T_c^1 can only play max_c games in the specified subset of slots S_c, against the teams j in T_c^2. CA2 constraints also contain an extra mode, HA, which considers the total amount of games played (both home and away).

$$\sum_{k \in S_c} \sum_{j \in T_c^2} (x_{i,j,k} + x_{j,i,k}) \leq max_c \qquad \forall c \in CA2^{HHA}, i \in T_c^1 \tag{4.15}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{i,j,k} \leq max_c \qquad \forall c \in CA2^{HH}, i \in T_c^1 \tag{4.16}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{j,i,k} \leq max_c \qquad \forall c \in CA2^{HA}, i \in T_c^1 \tag{4.17}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} (x_{i,j,k} + x_{j,i,k}) - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SHA}, i \in T_c^1 \tag{4.18}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{i,j,k} - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SH}, i \in T_c^1 \tag{4.19}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{j,i,k} - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SA}, i \in T_c^1 \tag{4.20}$$

Constraints (4.15-4.17) model the hard version of the three different modes HA, H, and A in CA2, while constraints (4.18-4.20) define the deviations, which equal the number of games (HA, H or A) more than max_c.

CA3 states that each team i in T_c^1 plays at most max_c home games, away games, or games against teams j in T_c^2, in each sequence of $intp_c$ time slots. E.g., Team 0 (from T_c^1) plays at most two consecutive matches against Team 1, 2, and 3 (from T_c^2) in each sequence of 3 ($intp_c$ time slots). Eg., if a match $x_{0,1,1}$ (Team 0 plays at home against Team 1 in round 1), then Team 0 can only play one more game against Team 2 or 3, in the following two rounds (sequence of 3).

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} (x_{i,j,l} + x_{j,i,l}) \leq max_c \tag{4.21}$$

$$\forall c \in CA3^{HHA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{i,j,l} \leq max_c \tag{4.22}$$

$$\forall c \in CA3^{HH}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{j,i,l} \leq max_c \tag{4.23}$$

$$\forall c \in CA3^{HA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} (x_{i,j,l} + x_{j,i,l}) - max_c \leq d_{i,k,c}^{CA3} \tag{4.24}$$

$$\forall c \in CA3^{SHA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{i,j,l} - max_c \leq d_{i,k,c}^{CA3} \tag{4.25}$$

$$\forall c \in CA3^{SH}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp^{CA3^S}-1} x_{j,i,l} - max_c \leq d_{i,k,c}^{CA3} \tag{4.26}$$

$$\forall c \in CA3^{SA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

Constraints (4.21-4.23) forces each team i in T_c^1 to play at most max_c HA/H/A games against teams j in T_c^2 for each sequence of $intp_c$ time slots, while (4.24-4.26) count the number of games (HA, H or A) that exceeds max_c number of games.

As mentioned earlier, CA4 is divided into modes referred to as *global* and *every*. For given subsets of teams i in T_c^1 and k in T_c^2, the *global* mode is used to limit the total number of games in the tournament, while the *every* mode limits the total number of games during each slot in a given subset of S_c. E.g., if we have a subset of rounds (1,2,3), *global* mode limits the total number of games in the tournament, while *every* limits the number of games in each of the rounds 1, 2, and 3.

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{k \in S_c} (x_{i,j,k} + x_{j,i,k}) \leq max_c \qquad \forall c \in CA4g^{HHA} \tag{4.27}$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{k \in S_c} x_{i,j,k} \leq max_c \qquad \forall c \in CA4g^{HH} \tag{4.28}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c} x_{j,i,k} \leq max_c \qquad \forall c \in CA4g^{HA} \tag{4.29}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c} (x_{i,j,k} + x_{j,i,k}) - max_c \leq d_c^{CA4g} \qquad \forall c \in CA4g^{SHA} \tag{4.30}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c} x_{i,j,k} - max_c \leq d_c^{CA4g} \qquad \forall c \in CA4g^{SH} \tag{4.31}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c} x_{j,i,k} - max_c \leq d_c^{CA4g} \qquad \forall c \in CA4g^{SA} \tag{4.32}$$

Equations (4.27-4.29) forces teams i in T_c^1 to play at most max_c HA/H/A games against teams j in T_c^2 during time slots S_c. Equations (4.30-4.32) counts the number of games (HA/H/A) more than max_c.

For mode *every* we have the following constraints:

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} (x_{i,j,k} + x_{j,i,k}) \leq max_c \qquad \forall c \in CA4e^{HHA}, \forall k \in S_c \tag{4.33}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} x_{i,j,k} \leq max_c \qquad \forall c \in CA4e^{HH}, \forall k \in S_c \tag{4.34}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} x_{j,i,k} \leq max_c \qquad \forall c \in CA4e^{HA}, \forall k \in S_c \tag{4.35}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} (x_{i,j,k} + x_{j,i,k}) - max_c \leq d_{k,c}^{CA4e} \qquad \forall c \in CA4e^{SHA}, k \in S_c \tag{4.36}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} x_{i,j,k} - max_c \leq d_{k,c}^{CA4e} \qquad \forall c \in CA4e^{SH}, \forall k \in S_c \tag{4.37}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2} x_{j,i,k} - max_c \leq d_{k,c}^{CA4e} \qquad \forall c \in CA4e^{SA}, \forall k \in S_c \tag{4.38}$$

Constraints (4.33-4.35) force teams i in T_c^1 to play at most max_c games (HA/H/A) against teams j in T_c^2 during each time slot k in S_c. The latter three constraints (4.36-4.38) count the number of games (HA/H/A) more than max_c, for teams in T_c^1 against opponents in T_c^2, during each time slot in S_c.

Game Constraints

GA1 specifies that at least min_c and at most max_c games from G_c take place during time slots k in S_c. Where G_c is a multiset of games that consist of ordered pairs *(i,j)* in which i is the home team providing the venue where the game is played, and j is the away team. E.g., G(0,1) cannot take place during Slot 2, as there are other major events on/around the venue of Team 0. On the other hand, G(0,1) must be played on Slot 2 due to broadcasting requirements.

$$min_c \leq \sum_{(i,j) \in G_c} \sum_{k \in S_c} x_{i,j,k} \qquad \forall c \in GA1^H \tag{4.39}$$

$$\sum_{(i,j) \in G_c} \sum_{k \in S_c} x_{i,j,k} \leq max_c \qquad \forall c \in GA1^H \tag{4.40}$$

$$min_c - \sum_{(i,j) \in G_c} \sum_{k \in S_c} x_{i,j,k} \leq d_c^{GA1} \qquad \forall c \in GA1^S \tag{4.41}$$

$$\sum_{(i,j) \in G_c} \sum_{k \in S_c} x_{i,j,k} - max_c \leq d_c^{GA1} \qquad \forall c \in GA1^S \tag{4.42}$$

The two constraints (4.39,4.40) concerns upper- and lower bounds for the hard version of the game constraints. The constraints force at least min_c and at most max_c games from the multiset G_c to take place during time slots S_c. The latter two equations (4.41,4.42) concern the soft versions of the constraints and trigger a deviation for the sum of games from G_c outside the interval of max_c - min_c for time slots S_c.

Break Constraints

BR1 can be used to avoid breaks at the beginning (end) of the season and limit the total number of breaks for each team. The following constraints (4.43-4.45) forces each team i in T_c^1 to have maximum $intp_c$ breaks (HA, H or A) during k slots in S_c

$$\sum_{k \in S_c} (h_{i,k} + a_{i,k}) \leq intp_c \qquad \forall c \in BR1^{HHA}, i \in T_c^1 \tag{4.43}$$

$$\sum_{k \in S_c} h_{i,k} \leq intp_c \qquad \forall c \in BR1^{HH}, i \in T_c^1 \tag{4.44}$$

$$\sum_{k \in S_c} a_{i,k} \leq intp_c \qquad \forall c \in BR1^{HA}, i \in T_c^1 \tag{4.45}$$

$$\sum_{k \in S_c} (h_{i,k} + a_{i,k}) - intp_c \leq d_{i,c}^{BR1} \qquad \forall c \in BR1^{SHA}, i \in T_c^1 \tag{4.46}$$

$$\sum_{k \in S_c} h_{i,k} - intp_c \leq d_{i,c}^{BR1} \qquad \forall c \in BR1^{SH}, i \in T_c^1 \tag{4.47}$$

$$\sum_{k\in S_c} a_{i,k} - intp_c \leq d^{BR1}_{i,c} \qquad \forall c \in BR1^{SA}, i \in T^1_c \tag{4.48}$$

Constraints (4.46-4.48) count the deviations, which occurs when a team in T^1_c have more than $intp_c$ breaks (HA, H or A) during the round(s) defined in S_c.

BR2 sums over all breaks (HA the only mode we consider) for teams i in T, which should be less or equal to $intp_c$ during time slots k in P. Constraint (4.49) puts a restriction on the total number of breaks in the season, while constraint (4.50) counts the total number of breaks exceeding $intp_c$.

$$\sum_{i\in T}\sum_{k\in P}(h_{i,k} + a_{i,k}) \leq intp_c \qquad \forall c \in BR2^{HHA} \tag{4.49}$$

$$\sum_{i\in T^1_c}\sum_{k\in S_c}(h_{i,k} + a_{i,k}) - intp_c \leq d^{BR2}_c \qquad \forall c \in BR2^{SHA} \tag{4.50}$$

Fairness Constraints

FA2, states that the number of home games (the only mode we consider) played by any two teams up until a given round should not exceed a given maximum. Thereby keeping a balance between the number of home games played in each round, for each pair of teams. Constraint (4.51) forces each pair of teams *(i,j)* in team group T^1_c to have a difference of played home games that is not larger than $intp_c$ after each k in S_c.

$$\sum_{0\leq l\leq k}\sum_{h\in T^1_c}(x_{i,h,l} - x_{j,h,l}) \leq intp_c \qquad \forall c \in FA2^H, k \in S_c, i,j \in T^1_c : i \neq j \tag{4.51}$$

$$\sum_{0\leq l\leq k}\sum_{h\in T^1_c}(x_{i,h,l} - x_{j,h,l}) - intp_c \leq d^{FA2^1}_{i,j,c} \tag{4.52}$$

$$\forall c \in FA2^S, k \in S_c, i,j \in T^1_c : i < j$$

$$\sum_{0\leq l\leq k}\sum_{h\in T^1_c}(x_{j,h,l} - x_{i,h,l}) - intp_c \leq d^{FA2^2}_{i,j,c}$$

$$\forall c \in FA2^S, k \in S_c, i,j \in T^1_c : i < j$$

Constraint (4.52) makes each pair of teams in T^1_c to trigger a deviation equal to the largest difference in played home games more than $intp_c$ over all k in S_c.

Separation Constraints

Separation constraints regulate the number of rounds between two matches involving the same opponents, as well as regulating the symmetry of the timetable.

SE1, constraint (4.53), states that each pair (*i,j*) of teams in T_c^1 should have at least min_c time slots between two matches involving the same opponents.

$$\sum_{k=\bar{k}}^{\bar{k}+min_c} (x_{i,j,\bar{k}} - x_{j,i,\bar{k}}) \leq 1 \qquad \forall c \in SE1^H, k \in P, i,j \in T_c^1 : i < j \tag{4.53}$$

$$\sum_{k=\bar{k}}^{\bar{k}+n}(x_{i,j,\bar{k}} - x_{j,i,\bar{k}}) - 1 \leq d_{i,j,c,n}^{SE1} \qquad \forall n \in min_c, c \in SE1^S, k \in P, i,j \in T_c^1 : i < j \tag{4.54}$$

For constraint (4.54) each pair of teams (*i,j*) in T_c^1, should trigger a deviation equal to slots less than $n = 1..min_c$ for all consecutive mutual games.

Objective Function

The objective function aims to minimize the total penalty obtained from violated soft constraints with their corresponding weights, while still respecting all hard constraints. The parameter w_c represents the weighted cost of violating a soft constraint.

$$min \sum_{i\in T} \sum_{c\in CA1^S} w_1 d_{i,c}^{CA1} + \sum_{i\in T} \sum_{c\in CA2^S} w_2 d_{i,c}^{CA2} + \sum_{i\in T} \sum_{c\in CA3^S} w_3 d_{i,k,c}^{CA3} + \sum_{c\in CA4g^S} w_4 d_c^{CA4g} +$$
$$\sum_{i\in T} \sum_{k\in P} \sum_{c\in CA4e^S} w_5 d_{i,k,c}^{CA4e} + \sum_{c\in GA1^S} w_6 d_c^{GA1} + \sum_{i\in T} \sum_{c\in BR1^S} w_7 d_{i,c}^{BR1} + \sum_{c\in BR2^S} w_8 d_c^{BR2} +$$
$$\sum_{i,j\in T} \sum_{c\in FA2^S} w_9 d_{i,j,c}^{FA2} + \sum_{i,j\in T} \sum_{c\in SE1^S} w_{10} d_{i,j,c}^{SE1}$$

4.2 Solution Approach

4.2.1 Heuristic

It is well known, in the sports scheduling literature, that solving MILP models are hard and you will not always find a feasible solution, and Durán et al. (*forthcoming*) also claim that some of their instances could run for days without even finding a feasible solution. To deal with this, we implement a heuristic approach where we would like to run an approximate model. Within this heuristic approach, we have different strategies when searching for feasibility and optimality, which we explain in section 4.2.

The main idea of the CHAP is to break the overall problem into smaller sub-problems by eliminating variables in the iterations. A HAP, as it appears in the literature, fixes the home-away status for every team throughout the season. CHAP, on the other hand, only fixes the home-away status for a subset of the teams. Our CHAP approach is based on running multiple iterations with several cluster patterns, giving us the benefit of diversification. This results in us having a more general approach where there could be several clusters, which helps reduce the dimension of the problem.

CHAP enables us to eliminate many variables, e.g., if a team is scheduled to play at home in round 1, all variables including this team playing away in round 1 are eliminated. This ensures that some constraints are satisfied, particularly BR2, which is hard to work with when fixing the HAP for a team. For the implementation of CHAP, we include the following constraint (4.55).

$$\sum_{j \in C_t} x_{jik} = p_{itk} \qquad \forall i, t, k \tag{4.55}$$

$$p_{i,t,k} \begin{cases} 1 & \text{if team is to play against teams of set } C_t \text{ in round k} \\ 0 & \text{otherwise} \end{cases}$$

Where $p_{i,t,k}$ is a binary parameter representing the CHAP. C_t is a subset of teams, representing a cluster. In general, for $C_t \subset T$, the number of clusters is predetermined to be n.

4.2.2 Symmetrics

When dealing with breaks, it is well known in the sports scheduling literature and practices that schemes can be quite helpful, as they allow you to have fewer breaks in a tournament. Goossens and Spieksma (2012), show that some form of symmetry is present in 20 out of the 25 European football leagues in their survey of the schedules in the respective leagues. Moreover, Durán and M. Guajardo (2017) obtained schedules that eliminated double-round breaks when running the base formulation in combination with various symmetry-related constraints. When running with the French scheme, they were able to satisfy all constraints. Thus, we include the following schemes in some of the iterations.

In the *English scheme*, meetings in the last time slot of the first half are equal to the first meetings in the second half. The slot $n + k$ in the second half also corresponds to slot k in the first half for slot $k = 1, 2..., n - 2$, and can be incorporated in the model using the following constraints:

$$x_{i,j,|T|-2} = x_{j,i,|T|-1} \qquad \forall i \in T, j \in T : i \neq j \tag{4.56}$$

$$x_{i,j,k} = x_{j,i,k+|T|} \qquad \forall i \in T, j \in T, k \in S : i \neq j, k \leq |T| - 3 \tag{4.57}$$

When an iterations includes the *French Scheme*, meetings in the first and last time-slots are equal along with the meetings in slot $n - 1 + k$ and slot $t + 1$ where $k = 1, 2..., n - 2$, with the home advantages being inverted.

$$x_{i,j,0} = x_{j,i,|S|-1} \qquad \forall i \in T, j \in T : i \neq j \tag{4.58}$$

$$x_{i,j,k} = x_{j,i,k-1+(|S|/2)} \qquad \forall i \in T, j \in T, k \in S : i \neq j, k \geq 1, k \leq (|S|/2) - 1 \tag{4.59}$$

In a *mirrored* tournament, we start by creating a solution for the first half of the season. Then, the second half is created as close to identical to the first half, with the only difference being that the home-away status of the teams is reversed.

$$x_{i,j,k} = x_{j,i,k+card(T)-1} \qquad \forall i \in T, j \in T, k \in S : i \neq j, k \leq (|S|/2) - 1 \tag{4.60}$$

The *Inverted* scheme is based upon the same principle of first creating the first half of the season. Then, the second half of the season is created, but unlike the mirrored format, the meetings in the second half are in reversed order of the meetings in the first half.

$$x_{i,j,k} = x_{j,i,2*(|T|-3-k)} \qquad \forall i \in T, j \in T, k \in S : i \neq j, k \geq 1, k \leq (|S|/2) - 1 \tag{4.61}$$

It is important to note that the iterations with the symmetric schemes are not guaranteed to work, and there might arise some situations where they conflict with the other hard constraints in the problem instances. However, we believe it is worth searching within these logical conditions as it might leave out undesirable nodes in the search progress. Additionally, we also believe it is worthy because it is promising in terms of the number of breaks and it will cut the feasibility space. However, this may come with the cost of cutting the optimal solution.

Since it can be hard to even find a feasible solution, instead of the objective function originally presented in section 4.1, we also include some alternative objective functions. In the following paragraph 4.2.3, we are running with feasibility with the purpose of finding a solution that satisfies all constraints.

4.2.3 Minimize Breaks

In general, the break constraints are complicated, which is well known in the sports literature, e.g., Cocchi et al. (2018) and Durán and M. Guajardo (2017). During our preliminary test, we experienced some challenges when dealing with breaks directly as a hard constraint. Thus we relax that constraint to get to a decent number of breaks by optimizing this number. In the following parts, we explain our strategies for the minimization of breaks.

Minimize Consecutive Breaks

When the CA3 hard constraints are present, the set contains two logical constraints. *(i) In every sequence of 3 games, no team can play more than two home games. (ii) In every sequence of 3 games, no team can play more than two away games.* In our first computational attempt, we quickly realized that the combination of these two constraints was quite time-consuming, and therefore motivates what follows.

After some experimentation, we found that the CA3 hard constraint would be satisfied by avoiding two consecutive breaks, and some slight adjustments are made, before starting a new search process.

First, we exclude the troubling hard constraints and introduce the auxiliary variables, $hh_{i,k}$ and $aa_{i,k}$, that take the value of 1, if a team has two consecutive home or away breaks, respectively. 0 otherwise.

$$hh_{i,k} \begin{cases} 1 & \text{if team i plays has two consecutive home breaks ending in round k} \\ 0 & \text{otherwise} \end{cases}$$

$$aa_{i,k} \begin{cases} 1 & \text{if team i plays has two consecutive away-breaks ending in round k} \\ 0 & \text{otherwise} \end{cases}$$

We then introduce four new constraints that make $hh_{i,k}$ and $aa_{i,k}$ take the value of 1 if a team has two consecutive home or away breaks, respectively. 0 otherwise.

$$\sum_{j \in T}(x_{i,j,k_{-2}} + x_{i,j,k_{-1}} + x_{i,j,k}) - 2 \leq hh_{i,k} \qquad \forall i \in T, k \in P : k > 1 \tag{4.62}$$

$$\sum_{j \in T}(x_{i,j,k_{-2}} + x_{i,j,k_{-1}} + x_{i,j,k}) - 2 \leq aa_{i,k} \qquad \forall i \in T, k \in P : k > 1 \tag{4.63}$$

$$hh_{i,k} \in \{0,1\} \qquad \forall i \in T, k \in P \tag{4.64}$$

$$aa_{i,k} \in \{0,1\} \qquad \forall i \in T, k \in P \tag{4.65}$$

After implementing the changes presented above, we use the following objective function to minimize the total number of breaks until reaching an objective value of 0.

$$minimize \qquad \sum_{i \in U} \sum_{k \in P}(hh_{i,k} + aa_{i,k}) \tag{4.66}$$

Ultimately, the steps conducted in this subsection should help us resolve any problems where the CA3 hard constraints serve as the only bottleneck. It is worth noting that the situation is different for every instance and that this strategy might not work in every case. However, we believe that every step that could help us achieve feasibility faster should be considered.

Minimize Total Breaks

In addition to the CA3 hard constraints, satisfying the BR2 hard constraint can also be quite time-consuming. In this section, we describe our strategy when the BR2 constraint serves as the major bottleneck in the problem instances.

First, we remove the BR2 hard constraints and present the objective function below. In a

2RR tournament, home and away breaks are considered to be co-existing, meaning that there has to exist an away break for every home break. Therefore, we only minimize home breaks, as this might slightly help lower the computation time, as opposed to minimizing $(h_{i,k} + a_{i,k})$.

$$\textit{minimize} \qquad \sum_{i \in U} \sum_{k \in P} h_{i,k} \tag{4.67}$$

The intuition is that this can help solve some of the instances where it has proven difficult to reach the allowed number of breaks. However, all of the instances are unique, and there is a high chance that a conflict arises between the BR2 constraint and the rest of the hard constraints that can force the incumbent objective to move slowly or not at all. When trying to solve this, we perform four runs while minimizing $h_{i,k}$, with each run subjecting the model to one of the symmetric schemes presented earlier.

5 Computation and Implementation

In this section, we briefly describe the data used in the ITC 2021, along with the data processing. Additionally, we elaborate on decisions made when searching for optimality.

5.1 Data and Computation

In total, we used four computers (with Windows 10, version 20H2) to run our models. We had two laptops at our disposal with an AMD 2.10 GHz 8 threads, 8 GB RAM, and Intel 2.1 GHz 4 threads, respectively. Additionally, we made use of the VMware desktop as provided by NHH, where we both had an account limited to 4GB RAM, and a 2.5 GHz CPU with 2 threads.

In addition to the computers, we also used *NEOS* (Dolan, 2001) with solver *GUROBI* (Gurobi Optimization, LLC, 2021), which allows for multiple instances to be solved simultaneously . The server is completely free[3], but does however have some limitations. Every job submitted to NEOS is limited to 3 GB of RAM and a maximum run time of 8 hours. The server contains multiple computers with different versions of the Intel Xeon CPU with 2.2-2.8 GHz, limited to 4 threads.

NEOS is not able to handle complex run codes, thus we perform all of the heuristic computations on the original four computers at our disposal.

Data

ITC 2021 uses the data format from RobinX, where the problem instances and solutions are stored in XML format. XML is usually preferred over plain text-only file formats due to the structured way of storing data and still being human-readable. Moreover, the set-up cost to work with XML is low, as most programming languages provide parsers to read and write XML files.

Moreover, the data in all instances are artificial but are thought to reflect the structure of tournaments within different sports. One instance can e.g., reflect the English Premier League, where you have 20 teams and 38 time slots. Table 5.1 contains all of the 45

[3]Developed by Wisconsin Institute of Discovery at the University of Wisconsin, Madison. They also welcome contribution to keep the optimizations flowing and to keep their services as available and free as possible

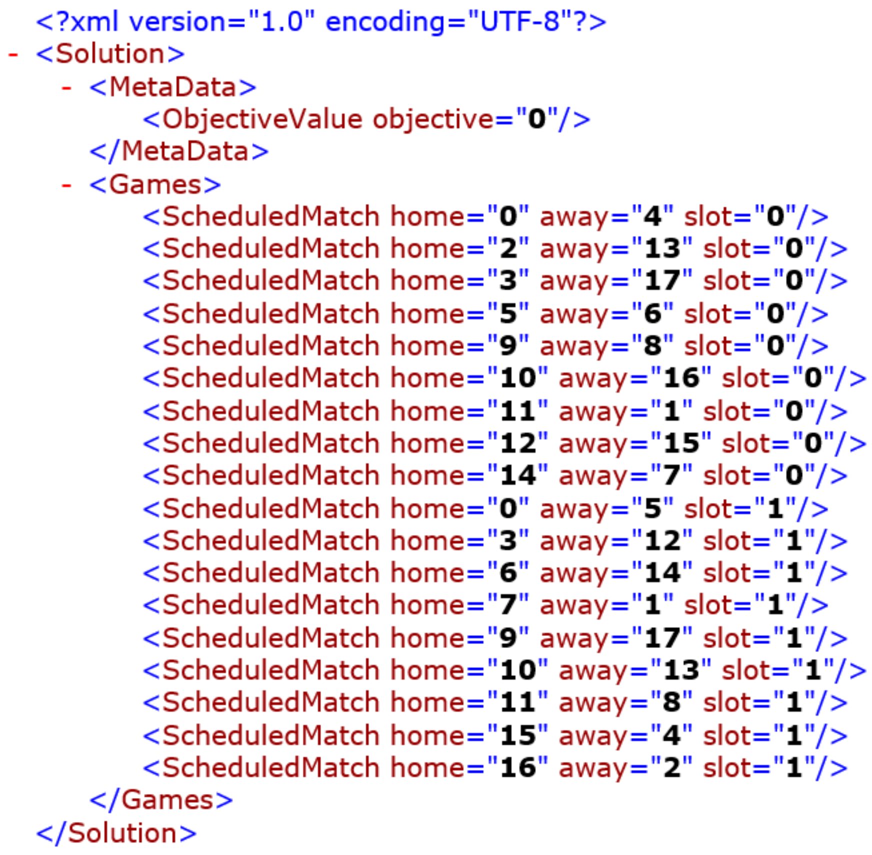
instances and gives an overview of the structure, types, and the number of constraints used in each of them.

Instance	Teams	Classification	Constraints
Early 1	16	P | BR1, BR2, CA1, CA2, CA4, FA2, GA1, SE1	206
Early 2	16	P | BR1, BR2, CA1, CA3, FA2, GA1	168
Early 3	16	P | BR1, BR2, CA1, CA2, CA3, FA2, GA1	335
Early 4	18	P | BR1, BR2, CA1, CA2, CA4, GA1, SE1	441
Early 5	18	P | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	803
Early 6	18	P | BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	999
Early 7	18	NULL | BR1, BR2, CA1, CA2, CA4, GA1, SE1	1343
Early 8	18	NULL | BR1, CA1, CA2, CA3, CA4, FA2, GA1	653
Early 9	18	NULL | BR1, BR2, CA1, CA2, CA3, FA2, GA1	193
Early 10	20	P | BR1, BR2, CA1, CA2, CA3, CA4, SE1	1270
Early 11	20	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1363
Early 12	20	P | BR1, BR2, CA1, CA2, CA3, CA4, GA1	214
Early 13	20	NULL | BR1, BR2, CA1, CA2, CA3, GA1	532
Early 14	20	NULL | BR1, BR2, CA1, FA2, GA1	113
Early 15	20	NULL | BR1, BR2, CA1, CA2, CA3, CA4, FA2, GA1	1412
Middle 1	16	P | BR1, BR2, CA1, CA2, CA4, SE1	1146
Middle 2	16	P | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1486
Middle 3	16	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1459
Middle 4	18	P | BR1, CA1, CA2, CA3, CA4, GA1	265
Middle 5	18	P | BR1, BR2, CA1, CA2, CA3, FA2, GA1	349
Middle 6	18	P | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	325
Middle 7	18	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	626
Middle 8	18	NULL | BR1, CA1, CA2, CA3, CA4, GA1	286
Middle 9	18	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1, FA2, GA1	296
Middle 10	20	P | BR1, BR2, CA1, CA2, CA4, GA1	912
Middle 11	20	P | BR1, CA1, CA2, CA4, GA1	1225
Middle 12	20	P | BR1, BR2, CA1, CA2, CA3, FA2, GA1, SE1	314
Middle 13	20	NULL | BR1, CA1, CA2, CA3, FA2, GA1, SE1	578
Middle 14	20	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	881
Middle 15	20	NULL | BR1, BR2, CA1, CA2, CA3, FA2, GA1, SE1	237
Late 1	16	NULL | BR1, CA1, CA2, CA3, CA4, FA2 GA1	778
Late 2	16	NULL | BR1, BR2, CA1, CA2, CA3, CA4, GA1	1323
Late 3	16	NULL | BR1, BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	576
Late 4	18	P | BR1, CA1, CA4, GA1, SE1	139
Late 5	18	P | BR2, CA1, CA2, CA3, CA4, FA2, GA1	924
Late 6	18	P | BR1, BR2, CA1, CA2, CA4, FA2, GA1, SE1	331
Late 7	18	NULL | BR1, BR2, CA1, CA2, CA3, GA1, SE1	873
Late 8	18	P | BR1, BR2, CA1, CA2, CA3, GA1, SE1	314
Late 9	18	NULL | BR1, BR2, CA1, CA2, CA3, FA2, GA1	505
Late 10	20	P | BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	936
Late 11	20	P | BR1, BR2, CA1, CA2, CA3, FA2, GA1	419
Late 12	20	NULL | BR1, BR2, CA1, CA2, CA3, CA4, SE1	1262
Late 13	20	NULL | BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	313
Late 14	20	NULL | BR1, CA1, CA2, CA3, CA4, FA2, GA1	1110
Late 15	20	NULL | BR1, BR2, CA1, CA3, FA2, GA1	93

Table 5.1: Overview of the structure, types and number of constraints

To ensure that our solutions are indeed feasible and that the objective values are computed are correctly, we use a Solution Validator provided by ITC 2021. The Validator also enable us to see the penalty obtained from each of the violated constraints.

At the end of *.run* files, we include some lines to write the solution output into *.csv*-files. Thus, we can paste the final timetables, from Excel, into the Validator. Subsequently, we generate our solution file (XML) and validate our computations. Figure 5.1, is an example of a solution file in XML format, where only the final timetable with the corresponding objective value had to be submitted, for each instance. Under *MetaData* we could have included additional information such as solution method and any remarks such as computational time.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <Solution>
  - <MetaData>
        <ObjectiveValue objective="0"/>
    </MetaData>
  - <Games>
        <ScheduledMatch home="0" away="4" slot="0"/>
        <ScheduledMatch home="2" away="13" slot="0"/>
        <ScheduledMatch home="3" away="17" slot="0"/>
        <ScheduledMatch home="5" away="6" slot="0"/>
        <ScheduledMatch home="9" away="8" slot="0"/>
        <ScheduledMatch home="10" away="16" slot="0"/>
        <ScheduledMatch home="11" away="1" slot="0"/>
        <ScheduledMatch home="12" away="15" slot="0"/>
        <ScheduledMatch home="14" away="7" slot="0"/>
        <ScheduledMatch home="0" away="5" slot="1"/>
        <ScheduledMatch home="3" away="12" slot="1"/>
        <ScheduledMatch home="6" away="14" slot="1"/>
        <ScheduledMatch home="7" away="1" slot="1"/>
        <ScheduledMatch home="9" away="17" slot="1"/>
        <ScheduledMatch home="10" away="13" slot="1"/>
        <ScheduledMatch home="11" away="8" slot="1"/>
        <ScheduledMatch home="15" away="4" slot="1"/>
        <ScheduledMatch home="16" away="2" slot="1"/>
    </Games>
  </Solution>
```

Figure 5.1: Solution file for best objective value obtained for L4. Showing all matches for the first two rounds

5.2 Implementation

As mentioned earlier, the data is stored in a structured and human-readable way. However, we still need to structure the data into the format needed for implementation in AMPL. We generate the default MILP model for each instance. When it comes to the different strategies for obtaining feasibility and optimality, we manually adjust these files and include the relevant constraints and/or objective functions for the respective instances.

In AMPL, three files are needed for each run, namely a *.mod*, a *.dat* and a *.run* file. The *.run* file remains more or less the same for each run, with some minor adjustments regarding the name of the model used in the run, input data, run time, and the number of partitions. While the *.mod* and .dat files are different for each instance and strategy.[4], which is why we create an R-script to generate all the *.dat* and *.mod* files needed to run the default MILP model for each instance. To read the XML files, we considered several parsers, such as *xml.etree.ElementTree* for Python, and *tinyxml2* for C++), but ended up using *XML* in *R*, in Rstudio (RStudio Team, PBC, 2021), as we have more experience with the programming language *R*.

After the parsing, the data is stored in a list of lists. Further, we extract the data for each type of constraint (CA1,..., CA4, GA1, BR1, etc.) and convert it into separate data frames, also referred to as tables. We still had some difficulties obtaining the data needed when stored like this. In order to extract the relevant data, we distinguish, not only between hard and soft constraints, but also between the different modes.(H, A, HA, mode *every, global* in CA4 etc.) If we take CA1 as an example, we filter this type of constraint into four new tables, as it has two modes, home (H) and away (A), and can be both soft and hard. To get a better picture of the filtering, we refer to how the different constraints are divided between different modes. This is the logic behind the filtering, which makes it easier for us to obtain the data we need from each constraint. Some of the constraints contain several teams or slots in a cell (like an excel cell). Thus, we extract the teams and slots into separate tables. E.g., $TCA1_1^{HH}$ contains the teams for the CA1 Hard (type) Home (mode) constraint 1, while $TCA1_2^{HH}$ contains the teams for CA1 Hard

[4]In the *.mod* files, the variables, and objective function remain the same when running the original MILP model in all instances. As mentioned, we also use different strategies where we make some adjustments to the default MILP by using slightly different objective functions and including some new variables

Home (mode) constraint 2, etc.

When the data is stored as intended, we create *for loops* that iterates through every row (constraint) and collects the data needed. All the sets are defined in the first loop, while the corresponding data are extracted in the second loop. The third loop defines all the constraints with the corresponding data. Moreover, the output from the first and third loop is stored in a .mod file, while the output from the second loop is store in a .dat file.

One of the main considerations when writing the R-script is that it should require few modifications between each instance. Thus, we write an R-scrip such that the only modifications needed between each run (between each instance) are the name of the XML file and changing to the corresponding working directory. Running the R-script, which resulted in the files needed for our runs AMPL, only took a couple of minutes (or less) per instance.

5.2.1 Clusters Patterns

In this section, we explain our strategies when searching for optimality.

When minimizing the original objective function, the feasible solutions serve as input for our mathematical solver while we search for optimality. However, simply altering the objective function and running the models are not likely to provide any remarkable results, as MILP problems in sports scheduling can take many weeks to solve (Klotz and Newman, 2013). Therefore, we need a method to speed up the process of finding a solution.

When limiting the search to a specific CHAP, the area left to search decreases significantly, and if we already have found a feasible solution, there might exist a better solution within the CHAPs of the teams. However, simply exploring the current CHAPs of the teams will most likely cut off many good solutions from the search area.

We start by creating partitions for the teams, where T, set containing all teams, is divided into three arbitrary groups. The partitions were created using a random permutation of the integer numbers 0 to 15 (17/19) in Rstudio [5]. Subsequently, we write CHAPs for the feasible solutions achieved. A pseudo-code describing Algorithm 1 is presented

[5]Using the sample() function and then ranking the random numbers using rank() from the programming language R.

below, where the cluster combination defines which cluster constraints are included in each iteration.

The algorithm starts from an initial feasible solution and then test running by fixing the CHAP in iterative runs. The difference from one to another run is given by different partitions and different combinations of the constraint (4.55) used to fix the CHAP. As we are using $n = 3$ clusters, we have three cluster constraints. Thus, we include the binary parameters $on1$, $on2$, and $on3$ to determine whether the respective cluster constraints are used in a run. E.g., cluster combination {1,1,1} implies that all three cluster constraints are used in the run, while {1,1,0} indicates that only cluster constraints 1 and 2 are included.

Algorithm 1

```
 1: procedure Minimize objective value
 2:     Write CHAP for the feasible solution;
 3:     for All Sets of Partitions do
 4:         for Cluster Combinations:{1,1,1},{1,1,0},{1,0,1},{0,1,1} do
 5:             Minimize Soft Constrains under Cluster Combination;
 6:             Run for model until proven optimally or expired time-limit;
 7:             if Incumbent Objective Improved then
 8:                 Update Initial Solution;
 9:                 Re-Write CHAP;
10:             end if
11:             Go to the next iteration
12:         end for
13:     end for
14:     Return Best solution obtained;
```

In total, only four combinations of the cluster constraints are used, as setting all the parameters equal to zero results in running the original model without the cluster constraints. Moreover, combinations with more zeroes than ones (e.g., on1=1, on2=0, on3=0) are not considered as it implies more freedom, and leads to a larger feasible space to search for candidate solutions[6]. The explicit *. run* file, including Algorithm 1, can be found in the Appendix 7.2.

There are numerous ways to experiment with Algorithm 1 above in terms of finding out what yields the best results. As the run with 12 partition sets made quite the impact, we expand to 85 partition sets while exploring with different cardinalities of sets within the

[6]Larger feasible space leads to a harder instance and might reduce the chance of finding better solutions in less than the 2 min (time per run)

partitions[7]. Furthermore, we experiment a bit with the amount of time allocated to each iteration. To better identify how an increase of time impacts the minimization process, we start by using only 5 seconds per iteration before increasing the time limit to 4 minutes per iteration.

[7]Different number of teams in each cluster

6 Results

In this section, we present the results from the MILP model presented in Section 4.1. We start by giving a summary of the best objective values obtained. Then, we provide a feature of the heuristic approach and a closer examination of the performance in the different runs. Lastly, we present our results from the strategies used to find feasible solutions.

6.1 Heuristic Results

As earlier mentioned, we test our heuristic approach on all of the 45 instances from the ITC 2021.

We start running the instances without an objective function, with the purpose of finding a feasible solution. Subsequently, we run the heuristic approach, starting from already feasible solutions, with the aim of reducing the objective values. Table 6.1 presents the best objective value achieved after running three heuristic- and two vanilla runs in each instance.

When scheduling the instances using the strategies presented in Chapter 4, we are able to obtain a feasible solutions for 40 out the 45 problem instances (Early 13/15, Middle 13/15, Late 14/15). The heuristic approach was also able to prove optimality for one of the instances (Late 4), reaching an objective value of 0.

Instance	Teams	Constraints	Best Objective
Early 1	16	206	1209
Early 2	16	168	461
Early 3	16	335	2245
Early 4	18	441	2360
Early 5	18	803	NA
Early 6	18	999	5660
Early 7	18	1343	7879
Early 8	18	653	5639
Early 9	18	193	5843
Early 10	20	1270	NA
Early 11	20	1363	7149
Early 12	20	214	1955
Early 13	20	532	414
Early 14	20	113	4884
Early 15	20	1412	6297
Middle 1	16	1146	NA
Middle 2	16	1486	NA
Middle 3	16	1459	12504
Middle 4	18	265	82
Middle 5	18	349	3888
Middle 6	18	325	3050
Middle 7	18	626	5010
Middle 8	18	286	371
Middle 9	18	296	2315
Middle 10	20	912	2024
Middle 11	20	1225	3868
Middle 12	20	314	4142
Middle 13	20	578	3001
Middle 14	20	881	1712
Middle 15	20	237	4363
Late 1	16	778	2987
Late 2	16	1323	6035
Late 3	16	576	3739
Late 4	18	139	0
Late 5	18	924	NA
Late 6	18	331	2436
Late 7	18	873	2638
Late 8	18	314	2843
Late 9	18	505	2220
Late 10	20	936	3256
Late 11	20	419	311
Late 12	20	1262	6916
Late 13	20	313	5641
Late 14	20	1110	2400
Late 15	20	93	6925

Table 6.1: Overview of Heuristic Results

6.1.1 Features of the heuristic computation

As mentioned earlier, the results presented in Table 6.1 come from multiple heuristic- and Vanilla runs. The *Vanilla-run* refer to running the original MILP model (as defined in Section 4.1) in NEOS. While working on the instances, we experience that it seemingly becomes harder to improve the objective value as we find better solutions, thus making it hard to draw any scientific conclusions based on the runs that are not using the same starting point.

Thus, we replicate the first heuristic run using a *vanilla* approach. Even though the heuristic in some occurrences uses a computation time of 96 minutes (48 iterations * 2 minutes), the realized time spent in computation, on average, ended at 68 minutes. The reduced computation time is due to the fact that some iterations are able to locate the best solution within the CHAP before the time limit expires. Thus, we set the time limit to 68 minutes for all the vanilla runs in order to get a better sense of how the heuristic method compares to a vanilla approach.

	Run 1 Heuristic	Run 1 Vanilla	Run 2 Heuristic	Run 3 Heuristic	Run 4 Vanilla
Avg. Time	68 min	68 min	21 min	168 min	480 min
Avg. Reduction	24.72%	9.52%	2.27%	10.07%	16.53%
(%)/Min	0.36%/min	0.14%/min	0.10%/min	0.05%/min	0.03%/min

Table 6.2: Comparison of all runs

As mentioned in Chapter 3, Durán et al. (*forthcoming*) found good solutions in a relatively short time frame when using a decomposition approach based on CHAP created from geographical attributes. Based on the ideas from this article, we decide to implement a similar CHAP approach to see how effective the approach will be when the clusters are completely randomized.

6.1.1.1 Closer Examination of the Heuristics

In this paragraph, we present a more detailed description of the results obtained from the heuristic runs.

The 1. Heuristic run is conducted using 12 partitions of CHAP combinations, where each iteration has a time limit of 2 minutes. Figure 6.1 illustrates the relationship between the average reduction for all instances and the number of partitions.

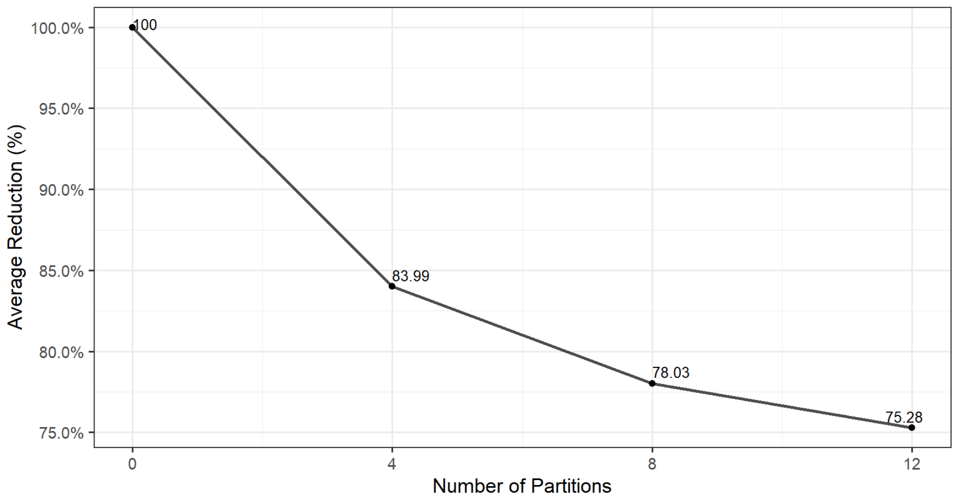


Figure 6.1: Average reduction for the first heuristic run, using 12 partitions and 2 minutes per iteration

The 1. Heuristic run is on average able to reduce the objective value by 24.7%. The reduction is also declining as the heuristic iterates over more of the partitions, suggesting that the minimization process gradually becomes harder as the objective improves.

The 1. Vanilla run presented in Table 6.2, is on average able to reduce the objectives by 9.52% with a computation time of 68 minutes. The 1. Heuristic runs, which use the same starting point, are able to make a reduction of 16.01% only using 4 partitions (approximately 16 minutes), meaning that a the heuristic approach outperforms the vanilla run in the minimization process.

The 2.- and 3. Heuristic runs are both conducted using 85 partitions. The difference to note between the runs is that the 2. run only uses 5 seconds per iteration, while the 3. run uses 4 minutes per iteration. Figure 6.2 illustrates the relationship between the average reduction in the objective function and the number of partitions for the 2.- and 3. run.

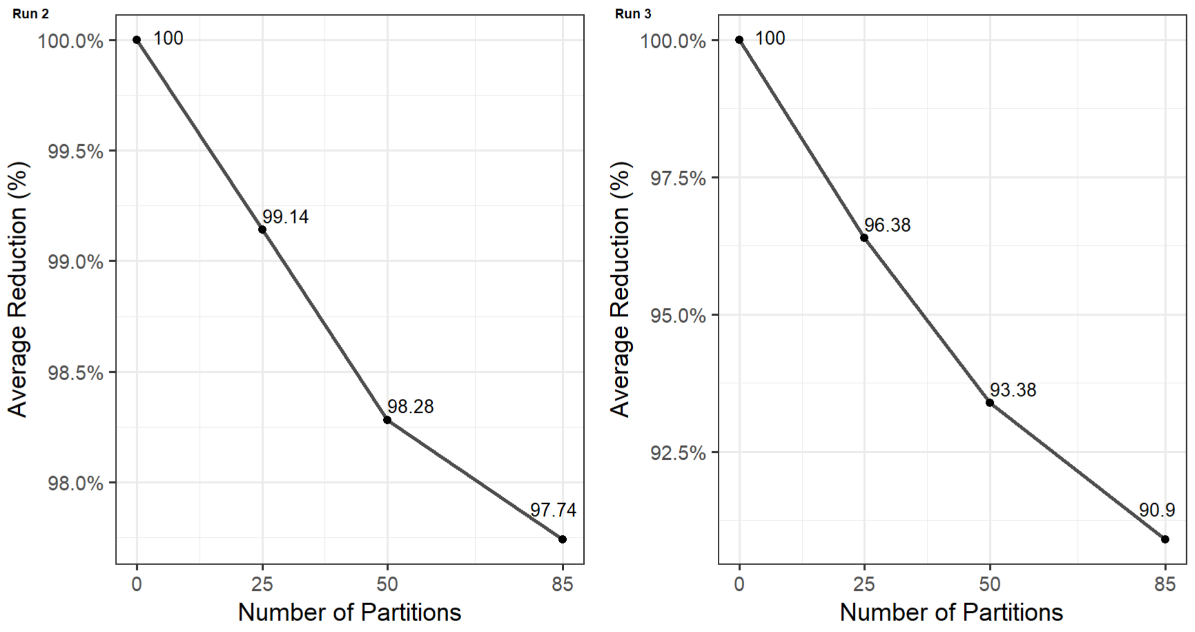


Figure 6.2: Average reduction for the 2.- and 3. heuristic run

As the starting values for the 2.- and 3. runs are different, we are not able to draw scientific conclusion based on this comparison. However, ther is an indication we would like to highlight. The 2. and 3. run, on average, decreases the objective value by 2.27% and 10.07%, respectively. It is interesting to see how well the 2. heuristic performs when only using 5 seconds per iteration, as opposed to 4 minutes, suggesting that the heuristic CHAP approach can work well, even with a low time limit.

6.2 Solution Approach

In Chapter 4, we presented our methodology for finding feasible solutions for the problem instances. Overall the level of difficulty varies across the instances. While some instances are solved in a matter of seconds, others would require multiple days of run time without being able to satisfy the hard constraints when using the vanilla approach.

As presented in the previous chapter, we start running without an objective function and use *min* 0 as a practicality in our code. Table 7.1 in the appendix gives an overview of the instances where a feasible solution has been found during an 8-hour run in NEOS without an objective function. During the 8 hour run, we achieve feasibility for 8/15 Early instances, 12/15 Middle instances, and 10/15 Late instances, overall achieving feasibility for 30/45 instances.

For the remaining 15 instances, we implemented the following strategies with the purpose of finding feasibility.

6.2.1 Minimizing Breaks

As earlier mentioned, we minimize consecutive breaks and the total number of breaks in order to satisfy the BR2 and CA3 hard constraints.

6.2.1.1 Minimize Consecutive Breaks

In total, there are 4 instances where we experience difficulties when trying to satisfy the CA3 hard constraints, which did not allow any teams to have consecutive breaks. Table 6.3 display the results after minimizing consecutive breaks ($hh_{i,k} + aa_{i,k}$), in NEOS with a time limit set to 8 hours. In the end, we manage to satisfy the CA3 hard constraints for two of the instances tested.

	Early 5	Early 11	Early 12	Middle 2
Early	3	0	0	13

Table 6.3: 8-hour consecutive break minimization in NEOS

As there are no other conflicting hard constraints in Early 12, we obtain feasibility for this instance based solely on this run. For the Early 11, we are able to reach zero consecutive breaks. However, as the requirement of the total number of breaks, BR2, are not met, the solution is kept and passed along to the break minimization process.

For the remaining instances Early 5, and Middle 2, we are not able to reach the criteria of zero consecutive breaks.

6.2.1.2 Minimizing Total Breaks

As previously mentioned, the BR2 constraint sets an upper bound on the total number of breaks in the tournament. Based on our experiences, the BR2 condition is the hardest one to satisfy out of the hard constraints.

Table 6.4 presents the results when minimizing breaks. The upper bound of the BR2 hard constraint is denoted in the parenthesis. The runs are conducted in NEOS using the four schemes presented in Chapter 4. The NA denotes the occurrences where the model did not find any feasible solutions during an 8-hour run. The denotation I is used, if the hard constraints are incompatible with the schemes.

	No scheme	French	English	Mirrored	Inverted
Early 1 *(78)*	**56***				
Early 2 *(76)*	**60***				
Early 4 *(88)*	136	100	96	108	**80***
Early 5 *(88)*	NA	I	I	NA	NA
Early 10 *(96)*	160	NA	NA	NA	NA
Early 11 *(98)*	184	NA	NA	NA	I
Middle 1 *(78)*	NA	I	I	I	I
Middle 2 *(78)*	NA	NA	I	I	I
Middle 10 *(100)*	NA	I	**42***		
Late 2 *(78)*	NA	140	I	NA	NA
Late 5 *(88)*	196	NA	NA	NA	NA
Late 10 *(100)*	NA	NA	136	146	120
Late 11 *(78)*	148	I	**42***		
Late 12 *(98)*	194	NA	140	160	I

Table 6.4: Minimization of breaks under symmetric schemes

The method allows us to obtain feasibility for 5 out of 14 instances. However, the main takeaway from the table is how efficient it is to search inside the symmetric schemes for some of the instances in this process. For Middle 10 and Late 11, we manage to obtain a very low number of breaks, far within the upper bound of BR2 hard for the respective instances. Furthermore, for the instances with solutions exceeding the upper bounds, searching within the symmetric schemes provides better results for 54 % of the instances.

We are not able to obtain feasible solutions for all of the instances using this approach. However, we are able to lower the total number of breaks for several instances. In a last attempt to obtain feasibility for the infeasible instances, we use the best solution obtained, from Table 6.4, and use it as an initial solution in a heuristic break minimization process.

6.2.1.3 Heuristic

In a last attempt to satisfy the BR2 hard constraint, we use the heuristic CHAP approach to minimize the total number of breaks in the tournament. The best solution obtained in the previous section will serve as starting values for the variables. Table 6.4, presents the results when improving the number of breaks obtained in the last paragraph. This strategy enables us to obtain feasible solutions for an additional 4 instances, denoted by $*$.

	Starting Value	Results	Reduction (%)
Early 5	NA	NA	NA
Early 10	160	132	-17.50%
Early 11	172	**82***	-52.33%
Middle 1	NA	102	NA
Middle 2	NA	NA	NA
Late 2	140	**68***	-51.43%
late 5	196	152	-22.45%
Late 10	120	**96***	-20.00%
Late 12	140	**64***	-54.29%

Table 6.5: Heuristic break minimization, 40 minute iterations

When examining the 6.5, we see that the heuristic approach works well when a longer time limit is allocated to each iteration. This results in us having feasible results for 4 additional problems. For the instances that have a feasible starting point (when excluding BR2) prior to the heuristics, we see a major reduction in the total number of breaks, averaging a decrease of 31.13%. This gives us reason to believe that our algorithm could potentially perform well in break-minimization problems.

7 Conclusion

In this paper, we have outlined the problem of generating a schedule for sports tournaments and proposed our solution approach while testing it on the data instances of the ITC 2021.[8] As with most sports scheduling problems, the process of finding good solutions has been difficult as we consider many conditions regarding sporting fairness and criteria requested by stakeholders (e.g., clubs, broadcasters, government), resulting in many and perhaps conflicting constraints.

The main contribution of this thesis lies in the implementation of a heuristic solution approach using a combination of Mixed-Integer Linear Programming (MILP) and cluster patterns, referred to as CHAP[9]. Our CHAP approach is based on running multiple iterations with multiple cluster patterns, giving us the benefit of diversification. As the CHAP approach is relatively new in the field of sports scheduling, our work shows that the methodology can provide satisfactory sports schedules for experimental problem instances.

Although (Durán et al. *Forthcoming*) originally constructed the CHAPs based on geographical attributes, it is interesting to see the computational advantages that come with our solution approach, even though the clusters are completely randomized. Therefore, we believe it could be worthwhile to conduct further research on how a similar solution approach would perform in real-life problems, where machine learning algorithms can be applied in the process of defining team clusters.

While RobinX features the majority of the most common constraints in sports scheduling, this thesis can also serve as an informative paper for those who are eager to learn about the fundamentals of sports scheduling.

[8]As mentioned, each instance is thought to reflect a league/tournament within different sports
[9]Derived from HAP, *Home-Away Patterns*

References

Alarcón, F., Durán, G., Guajardo, M., Miranda, J., Muñoz, H., Ramírez, L., Ramírez, M., Sauré, D., Siebert, M., Souyris, S., Weintraub, A., and Wolf-Yad, R. (2017). Operations Research Transforms the Scheduling of Chilean Soccer Leagues and South American World Cup Qualifiers. *Interfaces*, 47(1):52–69.

Bartsch, T., Drexl, A., and Kröger, S. (2006). Scheduling the professional soccer leagues of austria and germany. *Computers and Operations Research*, 33(7):1907–1937.

Campbell, R. and Chen, D.-S. (1976). A minimum distance basketball scheduling problem. *Management Science in Sports*, 4:15–25.

Cocchi, G., Galligari, A., Nicolino, F. P., Piccialli, V., Schoen, F., and Sciandrone, M. (2018). Scheduling the italian national volleyball tournament. *INFORMS Journal on Applied Analytics*, pages 271–284.

Dolan, E. D. (2001). The neos server 4.0 administrative guide.

Durán, G., Guajardo, M., Gutiérrez, F., Marenco, J., Sauré, D., and Zamorano, G. (2021). Scheduling the main professional football league of argentina. *INFORMS Journal on Applied Analytics, Forthcoming.*

Durán, G., Guajardo, M., Miranda, J., Sauré, D., Souyris, S., Weintraub, A., and Wolf, R. (2007). Scheduling the chilean soccer league by integer programming. *Interfaces*, 37(6):539–552.

Durán, G., Guajardo, M., and Wolf-Yadlin, R. (2012). Operations research techniques for scheduling chile's second division soccer league. *Interfaces*, 42(3):273–285.

Durán, G. and M. Guajardo, D. S. (2017). Scheduling the south american qualifiers to the 2018 fifa world cup by integer programming. *European Journal of Operational Research*, 262(3):1109–1115.

Fiallos, J., Pérez, J., Sabillón, F., and Licona, M. (2010). Scheduling soccer league of honduras using integer programming. *IIE Annual Conference and Expo 2010 Proceedings*.

Forrest, D. and Simmons, R. (2006). New issues in attendance demand: The case of the english football league. *Journal of Sports Economics*, 7(3):247–266.

Goossens and Spieksma, D.R., F. (2012). Soccer schedule in europe: an overview. *Journal of Scheduling*, 15(5):641–651.

Goossens, D. (2018). Optimization in sports league scheduling: Experiences from the belgian pro league soccer. *Operations Research and Enterprise Systems*, 884:3–19.

Goossens, D. and Spieksma, F. (2009). Scheduling the belgian soccer league. *Interfaces*, 39:109–118.

Gurobi Optimization, LLC (2021). Gurobi optimizer reference manual.

Hausken, M. D., Anderson, H., Fagerholt, K., and Flatberg, T. (2013). Scheduling the norwegian football league. *International Transactions in Operational Research*, 20(1):59–77.

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

Klotz, E. and Newman, A. M. (2013). Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1):18–32.

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

Pollard, R. and Pollard, G. (2005). Long-term trends in home advantage in professional team sports in north america and england (1876-2003). *Journal of sports sciences*, 23:50–337.

Rasmussen, R. (2008). Scheduling a triple round robin tournament for the best danish soccer league. *European Journal of Operational Research*, 185:795–810.

Rasmussen, R. V. (2006). *Hybrid IP/CP Methods for Solving Sports Scheduling Problems*. PhD thesis, University of Aarhus.

Recalde, D., Torres, R., and Vaca, P. (2013). Scheduling the professional ecuadorian football league by integer programming. *Computers and Operations Research*, 40(10):2478–2484.

Recalde D, Torres R, V. P. (2013). Scheduling the professional ecuadorian football league by integer programming. *Computers and Operations Research*, 40(10):2478–2484.

RStudio Team, PBC (2021). Rstudio: Integrated development environment for r.

Van Bulck, D., Goossens, D., Beliën, J., and Davari, M. (2020a). Problem description and file format.

Van Bulck, D., Goossens, D., Schönberger, J., and Guajardo, M. (2020b). A three-field classification and unified data format for round-robin sports timetabling. *European Journal of Operatinal Research*, 280:568–580.

Appendix

7.1 Detailed Model Formulation

Sets

T : Set of all teams

T_c^1 : First indexed subset of teams for every constraint

T_c^2 : Second indexed subset of teams for every constraint

P : Set of all rounds

S_c : Indexed Subset rounds for every constraint

G_c : Indexed Set of pairs (i,j) for every constraint

C : Set of Constraints

$CA1 \subseteq C$: Capacity 1, Subset of Constraints

$CA2 \subseteq C$: Capacity 2,Subset of Constraints

$CA3 \subseteq C$: Capacity 3,Subset of Constraints

$CA4g \subseteq C$: Capacity 4, Mode = Global, Subset of Constraints

$CA4e \subseteq C$: Capacity 4,Mode = Every, Subset of Constraints

$GA1 \subseteq C$: Game 1, Subset of Constraints

$BR1 \subseteq C$: Break 1, Subset of Constraints

$BR2 \subseteq C$: Break 2,Subset of Constraints

$FA2 \subseteq C$: Fairness 2, Subset of Constraints

$SE1 \subseteq C$: Separation 1, Subset of Constraints

$CA1^{HH} \subseteq C$: Subset of CA1 Hard Constraints, Mode = Home

$CA1^{HA} \subseteq C$: Subset of CA1 Hard Constraints, Mode = Away

$CA1^{SH} \subseteq C$: Subset of CA1 Soft Constraints, Mode = Home

$CA1^{SA} \subseteq C$: Subset of CA1 Soft Constraints, Mode = Away

$CA2^{HH} \subseteq C$: Subset of CA2 Hard Constraints, Mode = Home

$CA2^{HA} \subseteq C$: Subset of CA2 Hard Constraints, Mode = Away

$CA2^{HHA} \subseteq C$: Subset of CA2 Hard Constraints, Mode = Home/Away

$CA2^{SH} \subseteq C$: Subset of CA2 Soft Constraints, Mode = Home

$CA2^{HA} \subseteq C$: Subset of CA2 Soft Constraints, Mode = Away

$CA2^{SHA} \subseteq C$: Subset of CA2 Soft Constraints, Mode = Home/Away

$CA3^{HH} \subseteq C$: Subset of CA3 Hard Constraints, Mode = Home

$CA3^{HA} \subseteq C$: Subset of CA3 Hard Constraints, Mode = Away

$CA3^{HHA} \subseteq C$: Subset of CA3 Hard Constraints, Mode = Home/Away

$CA3^{SH} \subseteq C$: Subset of CA3 Soft Constraints, Mode = Home

$CA3^{HA} \subseteq C$: Subset of CA3 Soft Constraints, Mode = Away

$CA3^{SHA} \subseteq C$: Subset of CA3 Soft Constraints, Mode = Home/Away

$CA4g^{HH} \subseteq C$: Subset of CA4 Hard Constraints, Mode1 = Home, Mode2 = Global

$CA4g^{HA} \subseteq C$: Subset of CA4 Hard Constraints, Mode1 = Away, Mode2 = Global

$CA4g^{HHA} \subseteq C$: Subset of CA4 Hard Constraints, Mode1 = Home/Away, Mode2 = Global

$CA4g^{SH} \subseteq C$: Subset of CA4 Soft Constraints, Mode1 = Home, Mode2 = Global

$CA4g^{HA} \subseteq C$: Subset of CA4 Soft Constraints, Mode1 = Away, Mode2 = Global

$CA4g^{SHA} \subseteq C$: Subset of CA4 Soft Constraints, Mode1 = Home/Away, Mode2 = Global

$CA4e^{HH} \subseteq C$: Subset of CA4 Hard Constraints, Mode = Home, Mode2 = Global

$CA4e^{HA} \subseteq C$: Subset of CA4 Hard Constraints, Mode = Away, Mode2 = Global

$CA4e^{HHA} \subseteq C$: Subset of CA4 Hard Constraints, Mode = Home/Away, Mode2 = Global

$CA4e^{SH} \subseteq C$: Subset of CA4 Soft Constraints, Mode = Home, Mode2 = Global

$CA4e^{HA} \subseteq C$: Subset of CA4 Soft Constraints, Mode = Away, Mode2 = Global

$CA4e^{SHA} \subseteq C$: Subset of CA4 Soft Constraints, Mode = Home/Away, Mode2 = Global

$GA1^{H} \subseteq C$: Subset of GA1 Hard Constraints

$GA1^{S} \subseteq C$: Subset of GA1 Soft Constraints

$BR1^{HH} \subseteq C$: Subset of BR1 Hard Constraints, Mode = Home

$BR1^{HA} \subseteq C$: Subset of BR1 Hard Constraints, Mode = Away

$BR1^{HHA} \subseteq C$: Subset of BR1 Hard Constraints, Mode = Home/Away

$BR1^{SH} \subseteq C$: Subset of BR1 Soft Constraints, Mode = Home

$BR1^{HA} \subseteq C$: Subset of BR1 Soft Constraints, Mode = Away

$BR1^{SHA} \subseteq C$: Subset of BR1 Soft Constraints, Mode = Home/Away

$BR2^{H} \subseteq C$: Subset of BR2 Hard Constraints

$BR2^{S} \subseteq C$: Subset of BR2 Soft Constraints

$FA2^{H} \subseteq C$: Subset of FA2 Hard Constraints

$FA2^{S} \subseteq C$: Subset of FA2 Soft Constraints

$SE1^{H} \subseteq C$: Subset of SE1 Hard Constraints

$SE1^{S} \subseteq C$: Subset of SE1 Soft Constraints

Variables & parameters

$x_{i,j,k}$: 1 if team i plays home against team j in round s, 0 otherwise

$h_{i,k}$: 1 if team i plays home break in round s, 0 otherwise

$a_{i,k}$: 1 if team i plays away break in round s, 0 otherwise

$d_{i,c}^{CA1}$: Number of deviations for every team in CA1 Soft constraints

$d_{i,c}^{CA2}$: Number of deviations for every team in CA2 Soft constraints

$d_{i,k,c}^{CA3}$: Number of deviations for every team in every sequence of CA3 Soft constraints

d_{c}^{CA4g} : Number of deviations in CA4g Soft constraints

$d_{i,k,c}^{CA4e}$: Number of deviations for round CA4e Soft constraints

d_{c}^{GA1} : Number of deviations in GA1

$d_{i,c}^{BR1}$: Number of deviations for every team in BR1 Soft constraints

d_{c}^{BR2} : Number of deviations in BR2 Soft constraints

$d_{i,j,c}^{FA2}$: Number of deviations for every team in FA2 Soft constraints

$d_{i,j,c}^{SE1}$: Number of deviations for every pair of teams in SE1 Soft constraints

max_c : Max amount of games played in constraint

min_c : Min amount of games played in constraint

$intp_c$: Sequence of slots in constraint

w_c : vector of weighted cost of broken constraints

Objective Function

$$min \sum_{i \in T} \sum_{c \in CA1^S} w_1 d_{i,c}^{CA1} + \sum_{i \in T} \sum_{c \in CA2^S} w_2 d_{i,c}^{CA2} + \sum_{i \in T} \sum_{c \in CA3^S} w_3 d_{i,k,c}^{CA3} + \sum_{c \in CA4g^S} w_4 d_c^{CA4g} +$$
$$\sum_{i \in T} \sum_{k \in P} \sum_{c \in CA4e^S} w_5 d_{i,k,c}^{CA4e} + \sum_{c \in GA1^S} w_6 d_c^{GA1} + \sum_{i \in T} \sum_{c \in BR1^S} w_7 d_{i,c}^{BR1} + \sum_{c \in BR2^S} w_8 d_c^{BR2} +$$
$$\sum_{i,j \in T} \sum_{c \in FA2^S} w_9 d_{i,c}^{FA2} + \sum_{i,j \in T} \sum_{c \in SE1^S} w_{10} d_{i,j,c}^{SE1}$$

Subject to

$$x_{i,i,k} = 0 \qquad \forall i \in T, k \in P \tag{7.1}$$

$$\sum_{j \in U} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall i \in T, k \in P \tag{7.2}$$

$$\sum_{k \in P} x_{i,j,k} = 1 \qquad \forall i, j \in T : i \neq j \tag{7.3}$$

$$\sum_{j \in T} (x_{i,j,k_{-1}} + x_{i,j,k}) - b_{i,k} \leq 1 \qquad \forall i \in T, k \in P : k > 0 \tag{7.4}$$

$$\sum_{j \in T} (x_{j,i,k_{-1}} + x_{j,i,k}) - a_{i,k} \leq 1 \qquad i \in T, k \in P : k > 0 \tag{7.5}$$

$$\sum_{k=1}^{|P|/2} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall k \in P, i, j \in T : i \neq j \tag{7.6}$$

$$\sum_{k=(|P|/2)+1}^{|P|} (x_{i,j,k} + x_{j,i,k}) = 1 \qquad \forall k \in P, i, j \in T : i \neq j \tag{7.7}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{i,j,k} \leq max_c \qquad \forall c \in CA1^{HH}, i \in T^1 : i \neq j \tag{7.8}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{j,i,k} \leq max_c \qquad \forall c \in CA1^{HA}, i \in T^1 : i \neq j \tag{7.9}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{i,j,k} - max_c \leq d_{i,c}^{CA1} \qquad \forall c \in CA1^{SH}, i \in T^1 : i \neq j \tag{7.10}$$

$$\sum_{k \in S_c} \sum_{j \in T} x_{j,i,k} - max_c \leq d_{i,c}^{CA1} \qquad \forall c \in CA1^{SA}, i \in T^1 : i \neq j \tag{7.11}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} (x_{i,j,k} + x_{j,i,k}) \leq max_c \qquad \forall c \in CA2^{HHA}, i \in T_c^1 \tag{7.12}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{i,j,k} \leq max_c \qquad \forall c \in CA2^{HH}, i \in T_c^1 \tag{7.13}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{j,i,k} \leq max_c \qquad \forall c \in CA2^{HA}, i \in T_c^1 \tag{7.14}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} (x_{i,j,k} + x_{j,i,k}) - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SHA}, i \in T_c^1 \tag{7.15}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{i,j,k} - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SH}, i \in T_c^1 \tag{7.16}$$

$$\sum_{k \in S_c} \sum_{j \in T_c^2} x_{j,i,k} - max_c \leq d_{i,c}^{CA2} \qquad \forall c \in CA2^{SA}, i \in T_c^1 \tag{7.17}$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} (x_{i,j,l} + x_{j,i,l}) \leq max_c \tag{7.18}$$

$$\forall c \in CA3^{HHA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{i,j,l} \leq max_c \tag{7.19}$$

$$\forall c \in CA3^{HH}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{j,i,l} \leq max_c \tag{7.20}$$

$$\forall c \in CA3^{HA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} (x_{i,j,l} + x_{j,i,l}) - max_c \leq d_{i,k,c}^{CA3} \tag{7.21}$$

$$\forall c \in CA3^{SHA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} x_{i,j,l} - max_c \leq d_{i,k,c}^{CA3} \tag{7.22}$$

$$\forall c \in CA3^{SH}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp^{CA3^S}-1} x_{j,i,l} - max_c \leq d_{i,k,c}^{CA3} \tag{7.23}$$

$$\forall c \in CA3^{SA}, i \in T_c^1, k \in P : k \leq |k| + 1 - intp_c$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{k \in S_c} (x_{i,j,k} + x_{j,i,k}) \leq max_c \qquad \forall c \in CA4g^{HHA} \tag{7.24}$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{k \in S_c} x_{i,j,k} \leq max_c \qquad \forall c \in CA4g^{HH} \tag{7.25}$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{k \in S_c} x_{j,i,k} \leq max_c \qquad \forall c \in CA4g^{HA} \tag{7.26}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c}(x_{i,j,k}+x_{j,i,k})-max_c\leq d_c^{CA4g}\qquad \forall c\in CA4g^{SHA} \tag{7.27}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c}x_{i,j,k}-max_c\leq d_c^{CA4g}\qquad \forall c\in CA4g^{SH} \tag{7.28}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}\sum_{k\in S_c}x_{j,i,k}-max_c\leq d_c^{CA4g}\qquad \forall c\in CA4g^{SA} \tag{7.29}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}(x_{i,j,k}+x_{j,i,k})\leq max_c\qquad \forall c\in CA4e^{HHA},\forall k\in S_c \tag{7.30}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}x_{i,j,k}\leq max_c\qquad \forall c\in CA4e^{HH},\forall k\in S_c \tag{7.31}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}x_{j,i,k}\leq max_c\qquad \forall c\in CA4e^{HA},\forall k\in S_c \tag{7.32}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}(x_{i,j,k}+x_{j,i,k})-max_c\leq d_{k,c}^{CA4e}\qquad \forall c\in CA4e^{SHA},k\in S_c \tag{7.33}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}x_{i,j,k}-max_c\leq d_{k,c}^{CA4e}\qquad \forall c\in CA4e^{SH},\forall k\in S_c \tag{7.34}$$

$$\sum_{i\in T_c^1}\sum_{j\in T_c^2}x_{j,i,k}-max_c\leq d_{k,c}^{CA4e}\qquad \forall c\in CA4e^{SA},\forall k\in S_c \tag{7.35}$$

$$min_c\leq\sum_{(i,j)\in G_c}\sum_{k\in S_c}x_{i,j,k}\qquad \forall c\in GA1^{H} \tag{7.36}$$

$$\sum_{(i,j)\in G_c}\sum_{k\in S_c}x_{i,j,k}\leq max_c\qquad \forall c\in GA1^{H} \tag{7.37}$$

$$min_c-\sum_{(i,j)\in G_c}\sum_{k\in S_c}x_{i,j,k}\leq d_c^{GA1}\qquad \forall c\in GA1^{S} \tag{7.38}$$

$$\sum_{(i,j)\in G_c}\sum_{k\in S_c}x_{i,j,k}-max_c\leq d_c^{GA1}\qquad \forall c\in GA1^{S} \tag{7.39}$$

$$\sum_{k\in S_c}(h_{i,k}+a_{i,k})\leq intp_c\qquad \forall c\in BR1^{HHA},i\in T_c^1 \tag{7.40}$$

$$\sum_{k\in S_c}h_{i,k}\leq intp_c\qquad \forall c\in BR1^{HH},i\in T_c^1 \tag{7.41}$$

$$\sum_{k\in S_c}a_{i,k}\leq intp_c\qquad \forall c\in BR1^{HA},i\in T_c^1 \tag{7.42}$$

$$\sum_{k\in S_c}(h_{i,k}+a_{i,k})-intp_c\leq d_{i,c}^{BR1}\qquad \forall c\in BR1^{SHA},i\in T_c^1 \tag{7.43}$$

$$\sum_{k\in S_c}h_{i,k}-intp_c\leq d_{i,c}^{BR1}\qquad \forall c\in BR1^{SH},i\in T_c^1 \tag{7.44}$$

$$\sum_{k\in S_c}a_{i,k}-intp_c\leq d_{i,c}^{BR1}\qquad \forall c\in BR1^{SA},i\in T_c^1 \tag{7.45}$$

$$\sum_{i \in T_c^1} \sum_{k \in S_c} (h_{i,k} + a_{i,k}) \leq intp_c \qquad \forall c \in BR2^{HHA} \tag{7.46}$$

$$\sum_{i \in T_c^1} \sum_{k \in S_c} (h_{i,k} + a_{i,k}) - intp_c \leq d_c^{BR2} \qquad \forall c \in BR2^{SHA} \tag{7.47}$$

$$\sum_{0 \leq l \leq k} \sum_{h \in T_c^1} (x_{i,h,l} - x_{j,h,l}) \leq intp_c \qquad \forall c \in FA2^H, k \in S_c, i, j \in T_c^1 : i \neq j \tag{7.48}$$

$$\sum_{0 \leq l \leq k} \sum_{h \in T_c^1} (x_{i,h,l} - x_{j,h,l}) - intp_c \leq d_{i,j,c}^{FA2^1}$$

$$\forall c \in FA2^S, k \in S_c, i, j \in T_c^1 : i < j \tag{7.49}$$

$$\sum_{0 \leq l \leq k} \sum_{h \in T_c^1} (x_{j,h,l} - x_{i,h,l}) - intp_c \leq d_{i,j,c}^{FA2^2} \tag{7.50}$$

$$\forall c \in FA2^S, k \in S_c, (i, j) \in T_c^1 : i < j \tag{7.51}$$

$$\sum_{k=\bar{k}}^{\bar{k}+min_c} (x_{i,j,\bar{k}} - x_{j,i,\bar{k}}) \leq 1 \qquad \forall c \in SE1^H, k \in P, i, j \in T_c^1 : i < j \tag{7.52}$$

$$\sum_{k=\bar{k}}^{\bar{k}+min_c} (x_{i,j,\bar{k}} - x_{j,i,\bar{k}}) - 1 \leq d_{i,c}^{SE1} \qquad \forall c \in SE1^S, k \in P, i, j \in T_c^1 : i < j \tag{7.53}$$

7.2 AMPL Heuristic Run-file

```
reset;
model insert.mod;
data subsets.dat;
data initial_sol.dat;

data d_sol_A_Cluster1.dat;
data d_sol_A_Cluster2.dat;
data d_sol_A_Cluster3.dat;

option solver gurobi;
option gurobi_options 'outlev=1 timelim=120 logfile=RunProgress.log';
option eexit -19999;

param bestOF;   #best objective value found so far
let bestOF:=100000000; #a very high number to start with
param currentOF;
param npartitions;
let npartitions:=69;   #Number of partitions tried
param partitionfile symbolic;
param cluster1patternfile symbolic;
param cluster2patternfile symbolic;
param cluster3patternfile symbolic;
param solreportfile symbolic;
param nsols;   #number of candidate solutions found
let nsols:=0;
for {ipar in 1..npartitions} { #start a cycle from instance i = 1 to instance i = ninst
        let partitionfile := "set_partition"& ipar &".dat";
        data (partitionfile);                        #input data of this file
        let cluster1patternfile  :="Cluster1Pattern_"& ipar&".dat";
        let cluster2patternfile  :="Cluster2Pattern_"& ipar&".dat";
        let cluster3patternfile  :="Cluster3Pattern_"& ipar&".dat";
### HERE RE-WRITING CLUSTER PATTERNS OF THE BEST SOLUTION
if (ipar >=1) then{ #ONLY NEEDED when >=1, because for ipar=1 it was given as input
reset data TuplasA_CLUSTER1, TuplasA_CLUSTER2,TuplasA_CLUSTER3;

###################### WIRITING CLUSTER AWAY PATTERN ##############
print "set TuplasA_CLUSTER1:=" > (cluster1patternfile);
print "set TuplasA_CLUSTER2:=" > (cluster2patternfile);
print "set TuplasA_CLUSTER3:=" > (cluster3patternfile);
for{i in T}{
  for{k in S}{
   for{j in TeamsCLUSTER1:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster1patternfile);
        }
      }#j
   for{j in TeamsCLUSTER2:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster2patternfile);
        }
      }#j
   for{j in TeamsCLUSTER3:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster3patternfile);
        }
      }#j
  }#for k
} #for i
print ";" > (cluster1patternfile);
print ";" > (cluster2patternfile);
print ";" > (cluster3patternfile);
}#if re-writting

data (cluster1patternfile);
data (cluster2patternfile);
data (cluster3patternfile);
```

```
########### FINISH RE-WRITING, NOW READY TO RUN

#### FIRST RUN WITH PATTERN CONSTRAINTS ON ALL CLUSTERS
        let on1:=1;  #constraint on cluster 1 used
        let on2:=1;  #constraint on cluster 2 used
        let on3:=1;  #constraint on cluster 3 used
        display bestOF,ipar,on1,on2,on3;
        solve;
       if (solve_result <> "infeasible") then {
        let currentOF:= soft_constraints;
        if(currentOF < bestOF) then{
                let bestOF:=currentOF;
                let nsols:=nsols+1;
                let solreportfile := "sol_variables"& nsols &".txt";
                display soft_constraints,x >(solreportfile);
                display a >(solreportfile);
                display h >(solreportfile);
### HERE RE-WRITING CLUSTER PATTERNS OF THE BEST SOLUTION SO FAR
        let cluster1patternfile  :="Cluster1Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster2patternfile  :="Cluster2Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster3patternfile  :="Cluster3Pattern_"& ipar&"_sol"& nsols &".dat";
reset data TuplasA_CLUSTER1, TuplasA_CLUSTER2,TuplasA_CLUSTER3;
###################### WIRITING CLUSTER AWAY PATTERN #############
print "set TuplasA_CLUSTER1:=" > (cluster1patternfile);
print "set TuplasA_CLUSTER2:=" > (cluster2patternfile);
print "set TuplasA_CLUSTER3:=" > (cluster3patternfile);
for{i in T}{
  for{k in S}{
   for{j in TeamsCLUSTER1:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster1patternfile);
        }
      }#j
   for{j in TeamsCLUSTER2:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster2patternfile);
        }
      }#j
   for{j in TeamsCLUSTER3:j<>i}{
      if(x[j,i,k] = 1) then{
        print "(",i, ",",k, ")" > (cluster3patternfile);
        }
      }#j
  }#for k
} #for i
print ";" > (cluster1patternfile);
print ";" > (cluster2patternfile);
print ";" > (cluster3patternfile);

data (cluster1patternfile);
data (cluster2patternfile);
data (cluster3patternfile);
########### FINISH RE-WRITING, NOW READY TO RUN AGAIN
        }#if bestOF
}# if infeasible

#### SECOND: RUN WITH PATTERN CONSTRAINTS ON CLUSTERS 1 AND 2
        let on1:=1;  #constraint on cluster 1 used
        let on2:=1;  #constraint on cluster 2 used
        let on3:=0;  #constraint on cluster 3 used
        display bestOF,ipar,on1,on2,on3;
        solve;
       if (solve_result <> "infeasible") then {
        let currentOF:= soft_constraints;
        if(currentOF < bestOF) then{
                let bestOF:=currentOF;
                let nsols:=nsols+1;
                let solreportfile := "sol_variables"& nsols &".txt";
                display soft_constraints,x >(solreportfile);
                display a >(solreportfile);
                display h >(solreportfile);
        ### HERE RE-WRITING CLUSTER PATTERNS OF THE BEST SOLUTION SO FAR
        let cluster1patternfile  :="Cluster1Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster2patternfile  :="Cluster2Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster3patternfile  :="Cluster3Pattern_"& ipar&"_sol"& nsols &".dat";
        reset data TuplasA_CLUSTER1, TuplasA_CLUSTER2,TuplasA_CLUSTER3;
        ###################### WIRITING CLUSTER AWAY PATTERN ##############
        print "set TuplasA_CLUSTER1:=" > (cluster1patternfile);
        print "set TuplasA_CLUSTER2:=" > (cluster2patternfile);
        print "set TuplasA_CLUSTER3:=" > (cluster3patternfile);
        for{i in T}{
          for{k in S}{
           for{j in TeamsCLUSTER1:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster1patternfile);
                }
              }#j
           for{j in TeamsCLUSTER2:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster2patternfile);
                }
              }#j
           for{j in TeamsCLUSTER3:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster3patternfile);
                }
              }#j
          }#for k
        } #for i
        print ";" > (cluster1patternfile);
        print ";" > (cluster2patternfile);
        print ";" > (cluster3patternfile);

        data (cluster1patternfile);
        data (cluster2patternfile);
        data (cluster3patternfile);
        ########### FINISH RE-WRITING, NOW READY TO RUN AGAIN
        }#if bestOF
}# if infeasible

#### THIRD: RUN WITH PATTERN CONSTRAINTS ON CLUSTERS 1 AND 3
        let on1:=1;  #constraint on cluster 1 used
        let on2:=0;  #constraint on cluster 2 used
        let on3:=1;  #constraint on cluster 3 used
        display bestOF,ipar,on1,on2,on3;
        solve;
       if (solve_result <> "infeasible") then {
        let currentOF:= soft_constraints;
        if(currentOF < bestOF) then{
                let bestOF:=currentOF;
                let nsols:=nsols+1;
                let solreportfile := "sol_variables"& nsols &".txt";
                display soft_constraints,x >(solreportfile);
                display a >(solreportfile);
                display h >(solreportfile);
        ### HERE RE-WRITING CLUSTER PATTERNS OF THE BEST SOLUTION SO FAR
        let cluster1patternfile  :="Cluster1Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster2patternfile  :="Cluster2Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster3patternfile  :="Cluster3Pattern_"& ipar&"_sol"& nsols &".dat";
        reset data TuplasA_CLUSTER1, TuplasA_CLUSTER2,TuplasA_CLUSTER3;
        ###################### WIRITING CLUSTER AWAY PATTERN #############
        print "set TuplasA_CLUSTER1:=" > (cluster1patternfile);
        print "set TuplasA_CLUSTER2:=" > (cluster2patternfile);
        print "set TuplasA_CLUSTER3:=" > (cluster3patternfile);
        for{i in T}{
          for{k in S}{
           for{j in TeamsCLUSTER1:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster1patternfile);
                }
              }#j
           for{j in TeamsCLUSTER2:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster2patternfile);
                }
              }#j
           for{j in TeamsCLUSTER3:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster3patternfile);
                }
              }#j
          }#for k
        } #for i
        print ";" > (cluster1patternfile);
        print ";" > (cluster2patternfile);
        print ";" > (cluster3patternfile);

        data (cluster1patternfile);
        data (cluster2patternfile);
        data (cluster3patternfile);
        ########### FINISH RE-WRITING, NOW READY TO RUN AGAIN
        }#if bestOF
}# if infeasible

#### FOURTH: RUN WITH PATTERN CONSTRAINTS ON CLUSTERS 2 AND 3
        let on1:=0;  #constraint on cluster 1 used
        let on2:=1;  #constraint on cluster 2 used
        let on3:=1;  #constraint on cluster 3 used
        display bestOF,ipar,on1,on2,on3;
        solve;
       if (solve_result <> "infeasible") then {
        let currentOF:= soft_constraints;
        if(currentOF < bestOF) then{
                let bestOF:=currentOF;
                let nsols:=nsols+1;
                let solreportfile := "sol_variables"& nsols &".txt";
                display soft_constraints,x >(solreportfile);
                display a >(solreportfile);
                display h >(solreportfile);
        ### HERE RE-WRITING CLUSTER PATTERNS OF THE BEST SOLUTION SO FAR
        let cluster1patternfile  :="Cluster1Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster2patternfile  :="Cluster2Pattern_"& ipar&"_sol"& nsols &".dat";
        let cluster3patternfile  :="Cluster3Pattern_"& ipar&"_sol"& nsols &".dat";
        reset data TuplasA_CLUSTER1 , TuplasA_CLUSTER2,TuplasA_CLUSTER3;
        ###################### WIRITING CLUSTER AWAY PATTERN ##############
        print "set TuplasA_CLUSTER1:=" > (cluster1patternfile);
        print "set TuplasA_CLUSTER2:=" > (cluster2patternfile);
        print "set TuplasA_CLUSTER3:=" > (cluster3patternfile);
        for{i in T}{
          for{k in S}{
           for{j in TeamsCLUSTER1:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster1patternfile);
                }
              }#j
           for{j in TeamsCLUSTER2:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster2patternfile);
                }
              }#j
           for{j in TeamsCLUSTER3:j<>i}{
              if(x[j,i,k] = 1) then{
                print "(",i, ",",k, ")" > (cluster3patternfile);
                }
              }#j
          }#for k
        } #for i
        print ";" > (cluster1patternfile);
        print ";" > (cluster2patternfile);
        print ";" > (cluster3patternfile);

        data (cluster1patternfile);
        data (cluster2patternfile);
        data (cluster3patternfile);
        ########### FINISH RE-WRITING, NOW READY TO RUN AGAIN
        }#if bestOF
}# if infeasible
}#ipar

display bestOF;
```

7.3 Tables

	Feasible solution found	Feasible Solution Not Found
Early	3	1
	6	2
	7	4
	8	5
	9	10
	13	11
	14	12
	15	
Middle	3	1
	4	2
	5	10
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
Late	1	2
	3	5
	4	10
	6	11
	7	12
	8	
	9	
	13	
	14	
	15	

Table 7.1: 8-hour run in Neos without objective function