FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

THE TIME-RELAXED TRAVELLING TOURNAMENT PROBLEM

Word count: 27.605

Alexander De Munster
Student Number : 01504713

Bram D'haenens
Student Number : 01502044

Supervisor: Prof. Dr. Dries Goossens

Master's Dissertation submitted to obtain the degree of:

Master in Business Engineering: Operations Management

Academic year: 2019-2020

UNIVERSITEIT GENT

Confidentiality Agreement

I declare that the content of this Master's Dissertation may be consulted and/or reproduced, provided that the source is referenced.

Student's name: Alexander De Munster

I declare that the content of this Master's Dissertation may be consulted and/or reproduced, provided that the source is referenced.

Student's name: Bram D'haenens

Preface

This duo master's dissertation was realized in order to obtain the Master of Science's degree in Business Engineering - 'Operations Management'.

We would like to express our gratitude for those who helped us realize this master's dissertation. First, we would like to thank our supervisor Prof. Dr. Dries Goossens and his assistant Mr. David Van Bulck for their support and guidance. We could always contact them with our questions and other problems. Additionally, we are very grateful for the work other researchers have already achieved and published in this field.

Finally, we would like to thank our family and friends for their support and understanding during this intensive period. We also like to thank them for proofreading this master's dissertation.

Alexander De Munster and Bram D'haenens
May 29, 2020

Impact on this master's dissertation by the corona measures

By the beginning of March 2020, this dissertation was already in a phase were the additional information that needed to be provided by the supervisors was limited. Also, since no collaboration from external actors (e.g. surveys or interviews) was required to perform the originally planned research, the normal course of action could be maintained.

The corona measures taken by the Belgian government had the most impact on the communication between the students and the supervisors. Due to this being a duo master's dissertation, communication between the two students plays an essential role in delivering a comprehensive report. Next to the internal communication, obtaining feedback from the supervisors also required additional effort. The corona measures did not allow for further face to face meetings with the supervisors.

The aforementioned communication issues could be countered using the current digital technologies. The supervisors provided feedback and additional information by email and video calls, while the internal communication between the students was mainly done by direct messaging and video calls. Luckily, the only impact of the corona measures were these communication issues, which only required some additional effort from all involved actors. This also means that the scope and results of this master's dissertation were not really impacted by the corona measures.

This preamble is drawn up in consultation between the students and the supervisors and is approved by both.

Contents

List of Abbreviations

ACO ant colony optimisation

ALNS adaptive large neighbourhood search

ANT HYP ant based hyper-heuristic

BMS basic match schedule

BRA Brazilian instance class

cf. confer

CIRC circular distance

CLONALG clonal selection algorithm

CON constant distance

CP constraint programming

CSM complete search method

DFS* combines iterative deepening A* with depth-first branch-and-bound

DRR double round-robin

e.g. exempli gratiā

etc. et cetera

FIFA Fédération Internationale de Football Association

GA genetic algorithm

GAL galaxy instance class

HAP home-away pattern

i.e. id est

IAAF International Association of Athletics Federations

IDA* iterative deepening A*

ILB independent lower bound

IP integer programming

K-RTTP time-relaxed travelling tournament problem with K byes per team

L. HYP new learning hyper-heuristic

MA memetic algorithm

MDRR mirrored double round-robin

MRCPSP multi-mode resource-constrained project scheduling problem

NBA National Basketball Association

nbv number of violations

NFL National Football League

NHL National Hockey League

NL National League

NP nondetermenistic polynomial time

PBSA population-based simulated annealing

RAIS$_{TTP}$ artificial immune system for relaxed TTP

RPBSA time-relaxed population-based simulated annealing

RTCDMP time-relaxed timetable constrained distance minimisation problem

RTTP time-relaxed travelling tournament problem

RTTSA time-relaxed travelling tournament simulated annealing

SCC shortest chain closure

SRR single round-robin

STGP sports teams grouping problem

TCDMP timetable constrained distance minimisation problem

TS composite-neighbourhood tabu search

TSP travelling salesman problem

TTP travelling tournament problem

TTSA travelling tournament simulated annealing

TUP travelling umpire problem

U23 under 23

Ub Upper bound on trip length

UEFA Union Européenne de Football Association

VNS-CS Variable Neighborhood Search Clustering Search

List of Tables

List of Figures

List of Algorithms

1 Introduction

Sports, the most important sideshow in the world is the title of a book written by German sports journalist Horst Peets (1960). 60 years later, this title remains relevant. It could even be said that the sports world has become a real industry. This dissertation will focus on one of the many timetabling problems related to this 'sideshow', namely the time-relaxed travelling tournament problem. The goal of this problem is to minimise the total travel time or travel distance in a competition by adapting the timetable.

In this first section, the importance and relevance of this problem will be discussed. The first paragraph will discuss the importance of timetabling or scheduling in a sports competition. The next paragraphs will then provide more insight in the importance of time-relaxed timetabling in sports competitions. This is followed by a discussion on the importance of travel time. The final paragraphs of this introduction will then clarify why sports related subjects are interesting for a master's dissertation.

Scheduling is an important aspect of every competition. This can be clarified using similar arguments as Kendall, Knust, Ribeiro, and Urrutia used to show the relevance of their annotated bibliography on the subject (2010). In professional sports leagues, revenue maximisation and logistic optimisation are key and the timetable needs to facilitate this. On the other hand non-professional leagues have different problems, e.g. the large number of tournaments and competitors, which also require coordination and logistical efforts. Timetabling consists of fixing a date and a venue for every single game of a competition. Good fixtures are an important aspect of maximising revenues, ensuring attractiveness and keeping the interest of the media and the fans. Schedules will affect the performance (positively or negatively) of every team in the competition. Multiple decision makers, constraints and objectives have to be taken into account while coming up with the best schedule, indicating the difficulty of the task (Kendall et al., 2010).

This paragraph will focus on the importance of time-relaxed timetabling and will show the need for distinguishing the time-constrained and time-relaxed situation. A time-relaxed schedule is a schedule where more timeslots are available than strictly needed to schedule all games. This differs from time-constrained scheduling where the number of timeslots is the minimum number of slots required to schedule all games (Schönberger, Mattfeld, & Kopfer, 2004). Non-commercial sports leagues attract less public attention, therefore, scheduling constraints coming from e.g. broadcasting rights and security forces are usually non-existing. However, venue availability is in general more limited, since most venues of non-professional teams are shared. Additionally, non-professional players usually have a lower availability, due to having other activities (e.g. family and work) as well (Van Bulck, Goossens, & Spieksma, 2019). These availability limitations support the claim that non-commercial sports leagues are not suited for time-constrained scheduling. Most contributions in the world of sports timetabling deal with time-constrained scheduling, however, time-relaxed timetabling still remains an important subject. Mainly because of its omnipresence in non-professional competitions, where flexibility is key, but also because some professional competitions like the NBA (Bean & Birge, 1980) or NHL (Costa, 1995) are in fact time-relaxed (Van Bulck & Goossens, 2020).

There exist numerous optimisation problems in the sports world (cf. Section 3.1) of which this dissertation will cover one, the Time-Relaxed Travelling Tournament Problem (RTTP). The goal of the RTTP is to minimise the total distance to be travelled by varying the schedule of a tournament. In the classic European scenario, for example the Belgian Jupiler Pro League, a team travels back to its home venue after every away game, even when two consecutive away games are scheduled. Consequently, the total distance to be travelled is independent of the schedule. In a rather small country like Belgium, with a maximum distance between two home venues of top tier teams (KV Oostende and KAS Eupen) of about 250 km, this remains possible. However, when the home venues of the teams are located further from each other, e.g. a distance of over 2200 km between the venues of Chilean first tier clubs Huachipato and Deportes Iquique, it is not always in the best interest of the teams to return home after every away game (Durán et al., 2007). By introducing the concept of away trips, i.e. a team that has multiple consecutive away games will stay on the road instead of returning home, the total distance to be travelled can be reduced. The next paragraphs will focus on the importance of travel time for the main stakeholders of sports competitions: the teams and their players, the fans, the umpires and the parents (and other volunteers).

The first (and maybe the most important) stakeholders are the teams and their players. They are the core subject of the RTTP (and the corresponding time-constrained problem (TTP)). This paragraph discusses the effects of travel on the team and players performance. Many researchers have already focused on the potential existence of a home advantage. Several factors, e.g. local crowds, familiarity with local conditions and travel time of the opponent, appear to support this claim (Du Preez & Lambert, 2007; Pollard, Prieto, & Gómez, 2017). Domestic air-travel is frequently used as an explanation for poor away-match performance, but it is much harder to statistically prove the impact of travel time on match performance. When separating travel effects by means of regression analyses on performance data from team sports, it is shown that travel related issues only explain 1 or 2% of the overall variance in match outcome (Duffield & Fowler, 2017). The lack of supporting data does not render the (time-relaxed) travelling tournament problem irrelevant, since limiting the travel time and distance for all teams will create some kind of fairness between them. Limiting the total distance to be travelled is certainly perceived as something positive by teams and their players since this plays a positive role on their own comfort and on the travel related costs that teams need to incur.

A second important stakeholder in all sports competitions are the fans. All games have to be scheduled at times and locations that are convenient for the fans, while limiting the distance to be travelled by fans is also desirable. Many U.S.-based teams apply the concept of away trips, but the story in Europe is somewhat different. In European competitions the total distance that has been travelled at the end of the season will always be the same, since the concept of away trips without returning home is absent. There are still guidelines to optimise the distance to be travelled during specific sub-periods, e.g. the English football fixtures during the Christmas period. The travel time during this period is minimised to avoid fans having to travel long distances during the holiday period, a time of the year that is characterised by bad weather conditions and sometimes limited public transportation offerings (Kendall & Westphal, 2013).

Limiting the distance to be travelled in these periods might be an incentive for fans to still come to the stadium. If your team plays close enough to its home venue, which often means playing a local rival, the fans will not stay away. In fact, empirical research shows that these so called 'derby' games have a predicted rise of about 14% on the size of the crowd as compared to non-derby games (Buraimo, Forrest, & Simmons, 2009).

Another essential group of actors in many professional and amateur sports competitions are the umpires or referees. The travelling umpire problem (TUP) has the same objective as the TTP, namely minimising the total distance to be travelled, but is subject to different constraints and is defined for different actors (umpires instead of teams). In TTP, distance is minimised by creating away trips, while distance minimisation in TUP is done by assigning the umpire crews to the venues which are closest to them. The two main constraints of the TUP state that one umpire crew should be assigned to each team at least once during the season, and that one umpire crew should not be assigned to a team too often in short succession (Trick, Yildiz, & Yunes, 2012). The existence of sufficient research on the TUP confirms that the travel time and distance of umpire crews is an important factor in sports scheduling.

The previous paragraphs mainly cover research in a professional context and while this might be the most important in terms of economic value, this is certainly not the most important in terms of size. Only a fraction of all people participating in sports can be called professional players. Therefore, it is important to highlight the relevance of travel time minimisation for non-professional sports leagues. In these leagues, like in professional soccer leagues in Europe, the visitor teams always go to the game's venue and return home immediately after the game. This means that for a fixed league of teams, the distance is independent of the sequence of the games. Since the composition of recreational sports competitions, e.g. Belgian youth football leagues, is not solely based on the previous results of the competing teams, the composition of the leagues is not fixed when the games have to be scheduled. This characteristic introduces a third distance related problem: the sport teams grouping problem (STGP). The objective of the STGP is to divide all participating teams in different groups that each will form a separate competition while minimising the total distance to be travelled across all competitions (Toffolo, Christiaens, Spieksma, & Vanden Berghe, 2019). Compared to the TTP and the TUP, the STGP again comes with a different approach to the distance minimisation problem. Distance is minimised in the STGP by adapting the composition of the competition. The fact that the distance problem can be tackled from different viewpoints, again shows the importance of travel time in the sports world.

The final introductory paragraphs will cover the relevance of a sports related master's dissertation. All over the world, cities and countries make a bid to bring major sports events, like the Olympics or other worldwide and continental championships, to their city or country. These events provide additional jobs, improve the current infrastructure and create more tourism to the hosting city or country[1]. Additionally, they attract interest from all over the world and games are watched by millions of people at the same time. Therefore, a country like Qatar thinks it is interesting to invest more than $200 million in their bid

[1] https://edition.cnn.com/travel/article/tokyo-2020-olympics-rugby-world-cup-tourists/index.html

for the organisation of the 2022 FIFA World Cup[2].

A further illustration of how important sports are, can be found in the almost absurd amounts of money teams invest in their players. Merchandise and ticket sales go up when a world class player is signed[3], and if the player does what the team expects of him, the team will also perform better. A better team performance will further increase the fan base and often results in higher bonuses for the team (and the players). For example, the winner of the UEFA Champions League 2018/2019 could have earned over 60 million euros[4]. Another source of income for the clubs and players are the broadcasting rights. In the English Premier League, a total of more than 3 billion euros of TV money is divided amongst the top tier teams[5].

Apart from the previously listed economic factors, sports also play a big social role. Sports stadiums have evolved during the past decades. They are no longer just places where a few home games are played every month, the new sports stadiums have evolved into multi-functional areas with shops, offices and restaurants. This again shows that sports are relevant for the entire world, even when you do not practice sports or when you are not interested in them.

Even though aforementioned factors make sports a relevant study subject for a dissertation by themselves, one additional decisive characteristic needs to be mentioned. The fact that almost everyone can relate with sports makes it a real socially relevant subject. At some point, everyone gets involved with the subject and this makes problems related to sports somewhat easier to understand and more interesting. This and all previously mentioned socio-economic factors motivate researchers and students, like ourselves, to explore existing optimisation problems in the sports world.

The remainder of this paper is structured as follows. Section 2 covers the problem description and Section 3 delivers an overview of the existing literature related to the problem. The methodology of this dissertation is presented in Section 4 and tested in Section 5, where the experimental results are provided. Finally, the general conclusions are drawn and some subjects for future research are proposed in Section 6.

[2] https://bleacherreport.com/articles/1793593-how-qatar-won-the-right-to-host-the-2022-fifa-world-cup/

[3] https://www.cnbc.com/2018/07/18/juventus-sold-over-60-million-of-ronaldo-jerseys-in-just-one-day.html

[4] https://www.uefa.com/uefachampionsleague/news/newsid=2562033.html

[5] https://www.forbes.com/sites/bobbymcmahon/2019/08/04/premier-league-201920-odds-how-to-watch-changes/#2897209c13a1

2 Problem Description

2.1 Time-Relaxed Travelling Tournament Problem

In a t-round-robin tournament, every team plays every other team a fixed number t times during the competition. So in a double round-robin (or DRR), every team plays exactly two matches against every other team, one at their home venue and one at the home venue of the opponent. In a time-relaxed environment a competition has K timeslots more available than are required to schedule all matches. These K additional timeslots are filled with byes, i.e. timeslots where a team does not play a match.

The time-relaxed travelling tournament problem (K-RTTP) boils down to a double round-robin tournament in a time-relaxed environment (K byes per team) (Bao & Trick, 2010), characterised by,

- a set of n teams T, assume n to be even
- a set of timeslots (or rounds) R, with $|R| > 2(n-1)$ or $|R| = 2(n-1)+K$
- an $n \times n$ integer distance matrix D, where $D_{ij} (= D_{ji})$ equals the distance between team t_i and t_j, with $t_{i,j} \in T : i \neq j$

This means that every team has to play $2(n-1)$ matches during the competition. The output of the K-RTTP is a feasible double round-robin tournament schedule for n teams, consisting of $n(n-1)$ matches and nK byes scheduled over $2(n-1)+K$ rounds, which meets the following constraints.

- C1: Each team plays every other team exactly twice
- C2: Each team plays exactly once at the home venue of all other teams
- C3: Each team has a total of K byes
- C4: The length of all home stands and away trips should range from length L to length U ('atmost constraint') (Van Hentenryck & Vergados, 2007)
- C5: Two teams should not play each other twice on two consecutive rounds ('norepeat constraint') (Van Hentenryck & Vergados, 2007)

The resulting schedule is a combination of an opponent schedule and a home-away assignment. The opponent schedule, where each team is represented by a different integer and a bye is represented by a 0, determines the different matchups in every round. The home-away assignment shows for all rounds which teams will play at home (+) and which will play an away game (-). Byes are ignored in the home-away assignment.

Let us now consider the case where $t = 1$, $n = 4$ and $K = 1$, i.e. a single round-robin tournament (SRR) of 4 teams with 1 bye per team, as an example. It is clear that schedule S_1 (Table 1) can be created by combining the opponent schedule (Table 2) and the home-away assignment (Table 3). The home-away assignment can also be seen as a combination of the so-called home-away patterns (HAPs) of all teams, which are sequences of $n-1$ pluses and minuses and K zeros (Post & Woeginger, 2006). The HAP for team 1 in schedule S_1 can easily be found in Table 3 and is '-+0+'. Also, it is important to notice that there are no two teams with an equal HAP since every two teams have to play each other in one of the rounds.

T/R	1	2	3	4
1	-4	3	0	2
2	3	4	0	-1
3	-2	-1	4	0
4	1	-2	-3	0

Table 1: Schedule S_1 $(t = 1, n = 4, K = 1)$

T/R	1	2	3	4
1	4	3	0	2
2	3	4	0	1
3	2	1	4	0
4	1	2	3	0

Table 2: Opponent Schedule SRR $(t = 1, n = 4, K = 1)$

T/R	1	2	3	4
1	-	+	0	+
2	+	+	0	-
3	-	-	+	0
4	+	-	-	0

Table 3: Home-Away Assignment $(t = 1, n = 4, K = 1)$

Since the conflict between travel distance and the home/away patterns of the teams is the core of the K-RTTP, the total distance to be travelled in the tournament is used as the objective function, i.e. it can be used to compare different schedules. The HAP's of the teams are limited by the constraints proposed earlier in this section. In accordance with the literature, the following assumption, which considers the distance calculation, is added.

- Each team will start and end the competition at their home venue

This assumption implies that all teams that have an away game as final game, will still need to travel back to their home venue when the competition ends. This potential "post-season" travel is included in the distance calculation.

In this dissertation, a fair and optimal schedule minimises the total distance travelled during the tournament (for the tournament as a whole) while keeping the different HAP's compliant with the imposed constraints. In order to minimise or at least decrease the total distance to be travelled, the concepts of away breaks, away trips and home stands can be introduced. When team i has consecutive away games at the venues of team j and team k, team i can choose not to travel back to their own venue but travel directly from the venue of team j to the venue of team k. In this case team i will have an away break. When one or more consecutive away breaks occur, it can be called an away trip. These so called away trips can reduce the total distance that must be travelled by team i. The opposite of an away trip is the situation where a team plays several consecutive home games, this situation is called a home stand. The potential impact of away trips on the distance to be travelled by a team is made clear in the following example.

Schedule S_2 (Table 4) represents the complete double-round-robin schedule for a competition consisting of four teams $\{1, 2, 3, 4\}$. Each row represents the individual schedule of a team over all seven rounds, while each column represents the schedule of a single round. The presence of one bye game per team shows that this is an instance of the K-RTTP, $K = 1$. The distance matrix D (Table 5) is used to calculate the total distance travelled by a team. Notice that due to the pairwise distances, this matrix is symmetrical and has a 0-diagonal.

T/R	1	2	3	4	5	6	7
1	-4	0	-2	3	2	4	-3
2	3	4	1	4	-1	-3	0
3	-2	0	-4	-1	4	2	1
4	1	2	3	-2	-3	-1	0

Table 4: DRR Schedule S_2 $(n = 4, K = 1)$

i/j	1	2	3	4
1	0	3	5	4
2	3	0	4	5
3	5	4	0	3
4	4	5	3	0

Table 5: Distance Matrix D $(n = 4)$

As mentioned, the assumption that each team starts and ends the competition at their home venue needs to be included in the distance calculation. The distance travelled by team 1 is calculated by the following summation.

$$D_{14} + D_{44} + D_{42} + D_{21} + D_{11} + D_{11} + D_{13} + D_{31}$$

The first and last term take the starting and finishing of the competition at the home venue into account, while the second term indicates that the bye game in round 2 is spent on the road. Using the pairwise distances gives us a total distance to be travelled of 22 $(= 4 + 0 + 5 + 3 + 0 + 0 + 5 + 5)$. If teams would always return home after playing an away game, the distance travelled by team 1 could be calculated as follows:

$$D_{14} + D_{41} + D_{11} + D_{12} + D_{21} + D_{11} + D_{11} + D_{11} + D_{13} + D_{31}$$

This results in 24 $(= 4 + 4 + 0 + 3 + 3 + 0 + 0 + 0 + 5 + 5)$, which is higher than when team 1 would not return home after every away game.

2.2 Length of Home Stands and Away Trips

When working with a uniform instance i.e. an instance where the distance between any two stadiums is equal, maximising the number of breaks, minimises the number of trips in a tournament and therefore minimises the total distance to be travelled (Urrutia & Ribeiro, 2004). This supports the claim that an away break may reduce total travel distance. However, this break maximisation has its practical limitations. The length of an away trip, i.e. the number of consecutive away breaks, must be limited because of two main reasons.

Players can train and rest easier and probably better when they are at home. Also, long away breaks can result in feelings of homesickness with some players and this can affect their performance on the pitch. A second driver to limit the duration of away trips are the fans. Most fans will go watch the game when the match is at the home venue of their favourite team. It might be fun to go and see the game every weekend of a month, but nobody likes to wait three or four rounds until their team plays another home game (see e.g. Durán, Guajardo, and Sauré (2017)). Since both players and fans are a crucial part of all sports

tournaments, away trips will have a fixed upper bound on their length, even when this might affect the travel distance of the schedule in a negative way.

The limitations are applied by defining two integer parameters L and U. Parameter L sets a lower bound on the amount of consecutive home or away games that can be played. Usually $L = 1$, which makes an alteration of home and away games possible. The second integer parameter, U, defines an upper bound on the length of the home stands and away trips. Setting the value of U equal to $n - 1$ removes the limitation on the length of home stands and away trips. The lower bound of the travelled distance of a single team, which equals the optimal solution of the travelling salesman problem for visiting all home venues, can now always be reached. This scenario is theoretically possible in a time relaxed environment with $n * (n - 1)$ available timeslots where each game is booked in a different timeslot. In this scenario all teams could in turn play their $(n - 1)$ away games. This confirms the conjecture made by Bao and Trick (2010) that the ILB is tight for the K-RTTP when K is suffiently large. While this theoretically is the solution with the lowest total distance to be travelled, in practice limiting the upper bound parameter U (and the amount of byes K) is necessary, as previously mentioned.

The lower the value of U, the more frequent a team is required to travel to their home venue, which in turn could increase the total travelled distance. Note, however, that U should be larger than 1 (if $n > 2$). When no home stands or away breaks occur, the schedules alternate home and away games. In this scenario only two different schedules exist, one starting with a home game and one starting with an away game. Since no two teams can have exactly the same HAP (cf. supra), this type of competition can only exist for two teams. As soon as a third team comes in to play, no feasible schedule exists for $U = 1$.

2.3 K-RTTP Specifications

This dissertation discusses the K-RTTP for $K = \{1, 2, 3\}$. Since K is typically small, byes are ignored when defining home stands and away trips, and hence in the calculation of total distance travelled. The home-away pattern "+-0-" starts with one home game and is followed by an away trip with the length of two matches. The bye is spent on the road and no additional trips need to be considered while calculating the total distance travelled by the team following this pattern. Another advantage of setting the value of K relatively small is that solutions for the TTP are feasible for the K-RTTP for all $K \geq 0$ (and even, solutions of k_1-RTTP are feasible for k_2-RTTP, for $k_1 \leq k_2$) (Bao & Trick, 2010).

In accordance with previous literature on the (R)TTP, the values for U and L are set equal to respectively 3 and 1 and this dissertation will only cover problems with an even number of teams. If n is even and no byes occur in a round, this round will exist out of $n/2$ games. While n is even, a number of byes b can occur. In this scenario byes always come in pairs, so b will also be even. A round will then exist out of $(n - b)/2$ matches, while b teams will have a bye. Furthermore, the K-RTTP model in this dissertation complies with all 5 constraints mentioned in one of the previous paragraphs. Finally, byes can be placed in every timeslot, independent of it being the first or the last. Neither are bounds imposed on the number of consecutive byes. Teams are also able to spend byes spent on the road.

2.4 Problem Variants

This section will cover some of the existing variants on the K-RTTP that is covered in this dissertation. These variants can be obtained by imposing additional constraints considering the value of K, the symmetry of the schedule and the scheduling of byes.

2.4.1 Time-Constrained Travelling Tournament Problem

The first variant can actually be seen as a special case or specialisation of the K-RTTP. A concept B can be seen as a specialisation of a concept A if and only if: every instance of concept B is also an instance of concept A and instances of concept A which are not instances of concept B exist[6]. Since the time-constrained travelling tournament problem (TTP) covers the instances of the RTTP where $K = 0$ and solutions for the TTP are feasible for the K-RTTP for all $K \geq 0$, the TTP can clearly be seen as a special case of the RTTP. These time-constrained problems are defined by the fact that each team plays exactly one game in each round. A competition of n teams, n is even, consists out of $2*(n-1)$ rounds and in each round $n/2$ games are played. All other constraints, as well as the objective function, remain unchanged. The only difference is that some statements from the previous section become unnecessary since TTP is a time-constrained problem and byes do not occur, if n is even. Previous literature on the TTP shows that this problem is strongly NP-hard (Thielen & Westphal, 2011). Since the TTP is a specialisation of the K-RTTP it is logical to assume that the K-RTTP will at least match the level of complexity of the TTP, i.e. the K-RTTP is also strongly NP-hard. NP-hard problems are at least as hard to solve as the hardest NP-problems, i.e. problems that can currently not be solved in polynomial time (Garey & Johnson, 1979). Since complexity theory does not belong to the scope of this dissertation, it is sufficient to say that NP-hard problems require evaluation of an immense number of possibilities to determine the exact solution. Here, heuristics can effectively reduce the number of evaluations needed and play an invaluable role in obtaining solutions within a reasonable time frame (Pearl, 1984).

2.4.2 Mirrored Double Round-Robin Tournament

A second variant of the K-RTTP studied in this dissertation is the mirrored double round-robin tournament (MDRR). Here, the first $n-1$ matches are scheduled like a single round-robin tournament. The last $n-1$ matches can then easily be scheduled by flipping the venues of the first $n-1$ matches. Apart from the easy implementation, an advantage of the MDRR is that it guarantees that every team plays every other team once in both parts of the competition. This variant of the (R)TTP is called the (R)TTP/Mirror problem (Easton, Nemhauser, & Trick, 2001). All feasible schedules of the (R)TTP/Mirror are also feasible schedules for the (R)TTP.

[6] https://en.wikipedia.org/wiki/Specialization_(logic)

2.4.3 Restrictions on the Scheduling of Byes

Three additional variants can be defined by putting some restrictions on the scheduling of byes (Brandão & Pedroso, 2014). A first variant to the RTTP only allows byes to occur during home stands. Rounds without a game will always be spent at the home venue in this scenario. A second constraint can be added to increase the fairness of the competition, all teams are now required to play a game in the last time slot. By adding this constraint no team will end its matches before the others and therefore the possibility of matchfixing is limited. This second constraint can be further extended to make sure that all teams start their competition in the same round, i.e. no byes are allowed in the first round. Finally, the third variant discussed by Brandão and Pedroso (2014) combines the first two and limits the RTTP to only allowing byes during home stands and not allowing byes in the final round of the competition.

3 Literature Overview

This section starts with a brief overview of optimisation problems (Section 3.1) and scheduling problems (Section 3.2) in the sports world. This is followed by an overview of the literature on the applied move operators, algorithms and heuristics of the TTP (Section 3.3). Next, the same subjects are covered for the RTTP (Section 3.4). Finally, some additional readings (Section 3.5) and a summary of applied move operators and experimental results in previous research (Section 3.6) are presented.

3.1 Optimisation Problems in Sports

The sizeable socio-economic impact of sports is one of the drivers in the search for optimal solutions for all sports related problems. A large amount of optimisation problems are scheduling problems. These problems also include the time-relaxed travelling tournament problem, which is the subject of this master's dissertation. This section gives a short summary of some other optimisation problems that exist in the sports world. More details about scheduling problems are discussed in Section 3.2.

A first sports related optimisation problem is the sports selection problem, studied by Rogulj, Papić, and Čavala (2009). Children should practise the sport that fits them best. Optimisation of the initial choice of sports will help improve the results a person will achieve. Rogulj et al. (2009) state that different sports are determined by authentic kinesiological structures and specific anthropological characteristics. Success of an individual in particular sports activities seems to be predominantly determined by the compatibility of his/her anthropological characteristics with the anthropological model of top athletes in that sport. Emerging new information technologies and the introduction of new methods and knowledge create the possibility to develop a systematic and scientifically based approach in selecting the appropriate sport (Rogulj et al., 2009).

Due to an increase in competitiveness in most sports, teams and organisations will try to identify talented athletes at a very young age. By implementing an effective talent identification system the likelihood of success can be significantly increased and future financial rewards could also increase (e.g. young talents have a higher market value) (Mann, Dehghansai, & Baker, 2017). The importance of the problem is clear but the practical prediction of future talents still remains challenging. The fact that future performance is impacted by so many different variables, ranging from physical capability to psychological factors, makes this optimisation problem an interesting subject for further research.

Every sport has its own game defining moments, like e.g. penalty kicks in football or free throws in basketball. These situations have in common that they are all not that hard to execute. Every professional football player can aim the ball to one of the four corners from only eleven meters and every NBA-player can throw a ball in the hoop from fifteen feet away. Yet, in reality, football players launch their penalty kick into the crowd and basketball players ruin their free throws, regardless of the limited technical difficulty of these moments. There is a clear link between metabolic power and fatigue (heart rate is used as

an indicator) and the performance on free throws (Padulo et al., 2015). These findings can be used to determine which players will probably excel in these game defining moments.

While the three presented subjects show the variety of optimisation problems in the sports world. A large collection of additional optimisation subjects, all with their own relevant research, exists and includes e.g. batting orders in cricket (Norman & Clarke, 2010), Formula 1 car specifications (Wloch & Bentley, 2004) and timing of soccer substitutions (Myers, 2012).

3.2 Scheduling in Sports

On the topic of scheduling sports leagues, two different cases can be considered. First, there are the temporally constrained problems, where there are exactly enough timeslots to schedule all games. Secondly, there are the time-relaxed problems. From a theoretical perspective, time-relaxed scheduling problems usually have a larger set of possible outcomes since the restriction of each team playing in each round is removed. In this case, each team will have one or more rounds in which no game has to be played, even though the league consists of an even number of teams. In time-relaxed scheduling problems the number of rounds will be larger than the minimum number of rounds needed to schedule all games. This dissertation will mainly focus on time-relaxed scheduling problems.

A lot of real-life competitions (at least in Europe) take part in a double round-robin format, i.e. all teams play two games against all of their opponents, one home game and one away game (Goossens & Spieksma, 2012). This implies that for a competition of twenty teams, for example the English Premier League, a total of 380 games should be played and scheduled. Variations on this format of course exist, some tournaments use a single round-robin format, where each team plays every other team just once at a fixed location. A small real-life example of this can be found in the group stage of the FIFA World Cup. The round-robin format surely is not the only format used in real life. A different format can be found in the NBA (National Basketball Association) where teams play more often against the teams that play in the same conference and the same division. Even though a lot of variation is present in tournament scheduling, only double round-robin tournaments will be covered in this dissertation.

As mentioned in earlier sections, stadiums have become more than just the venue of a few home games of a single club each month. It could be as clear as two clubs sharing the same stadium for their home games, like for example Club Brugge K.V. and Cercle Brugge KSV in the Belgian Jupiler Pro League (Goossens, 2017) or FC Internationale Milano and AC Milan in Italian Serie A (Della Croce & Oliveri, 2006). This might seem like a minor problem, but it has serious implications since it is impossible for both teams to have a home game in the same weekend. It gets even more complicated when both teams play in another division, which was the case in Belgium from 2015 until 2018 when Club Brugge played in first division while Cercle Brugge played in second. This has the implication that two different leagues must be at least partially aligned.

Some venues are also available for more than one sports discipline, for example the King Baudouin Stadium which is used as the home venue for Belgium's national football team, but also facilitates the Memorial Van Damme, the IAAF Diamond League's final event. The 2019 Memorial Van

Damme was scheduled on a timeslot where UEFA EURO2020 qualifiers took place. The qualifier schedule had to be adapted causing the Belgian national football team to play two away games during this international break. Finally, some venues are also used as a location for concerts or other cultural activities. For example, the Wembley stadium, the home venue of the England national football team, also hosted concerts of e.g. P!NK and Bon Jovi in 2019. The multidisciplinary use of these stadiums may be a great development for the importance of sports, but it implies availability constraints on the venues of the teams.

An additional complication of the scheduling problem can be found in the fact that for many sports different competitions overlap. Teams play their domestic competition(s) and maybe a continental or even a world cup. Most teams want to perform at their best in each of these competitions and the organisers want to see all teams at their best, so the schedules should be aligned. An example of misalignment between several competitions could be witnessed in December 2019, when Liverpool FC had to play a game for the domestic Carabao Cup on the 17th, while having to play the semi-finals of the FIFA Club World Cup on the 18th. In the end, the club decided to let the first team compete in the World Cup, while leaving the Carabao Cup for the U23-squad[7].

Timetabling problems have been extensively researched in previous literature, e.g. yearly published papers on (R)TTP (Figure 1). Due to the abundance of research on the topic, this literature overview can and should only cover the literature relevant for this dissertation. For a more complete overview of the existing research on sports timetabling, the reader is referred to several general sports timetabling papers (Drexl and Knust (2007), Rasmussen and Trick (2008), Knust (2010), Van Bulck, Goossens, Schönberger, and Guajardo (2020)).

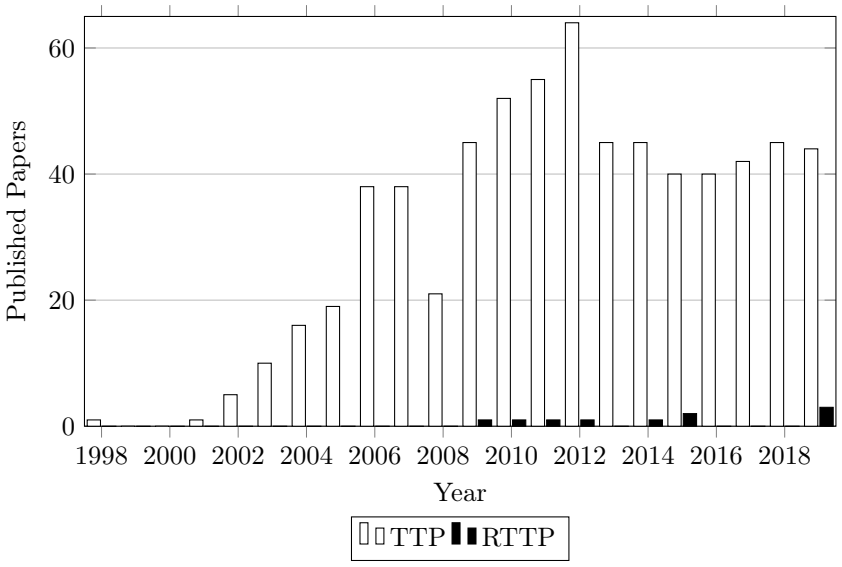


Figure 1: Published (R)TTP Papers

[7] https://www.bbc.com/sport/football/50827321

3.3 TTP

This section provides an overview of the TTP-literature relevant for this dissertation. More specifically, some interesting move operators (Section 3.3.1) and algorithms and heuristics (Section 3.3.2) are covered. The final part of this section discusses the construction of a TTP schedule (Section 3.3.3).

3.3.1 Move Operators

A move operator is defined as a mechanism that performs a certain move or change to an existing schedule in order to create a new schedule, which will also be part of the search space. This search space is defined as the set of schedules that can be obtained by applying any linear combination of move operators on any member of this set. All schedules that are part of the search space comply with the the first 3 constraints (C1, C2 and C3) of Section 2.1, later defined as the hard constraints. Schedules in the search space can violate the last 2 contraints (the atmost and norepeat constraint) of Section 2.1, later defined as soft constraints. The solution space of this dissertation is defined as the subset of the search space containing the schedules that comply with all five constraints of Section 2.1, members of the solution space will be called feasible schedules. For the time-constrained travelling tournament problem, Anagnostopoulos, Michel, Van Hentenryck, and Vergados (2006) have defined five basic move operators. These five operators, or a selection of them, are implemented in more than six methods that were created to tackle the TTP. When performing some minor alterations to these operators, they can also be implemented in RTTP-solving algorithms (Section 3.4.1).

3.3.1.1 Swap Homes

The first move operator is called *SwapHomes* and was proposed by Anagnostopoulos et al. (2006). It basically swaps the home and away roles of two teams, t_i and t_j, in a competition. If t_i and t_j have a match-up in round r_x and round r_y, where t_i plays at home in r_x and plays away in r_y, the move operator $SwapHomes(S, t_i, t_j)$ will change the schedule S such that t_i will play at home in r_y and away in r_x against team t_j. An example of this move operator can be found in Table 6.

T/R	1	2	3	4	5	6
1	4	3	-2	-3	2	-4
2	-3	-4	1	4	-1	3
3	2	-1	-4	1	4	-2
4	-1	2	3	-2	-3	1

T/R	1	2	3	4	5	6
1	4	3	-2	-3	2	-4
2	3	-4	1	4	-1	-3
3	-2	-1	-4	1	4	2
4	-1	2	3	-2	-3	1

Table 6: Example $SwapHomes(S, t_2, t_3)$

3.3.1.2 Swap Rounds

The second move operator proposed by Anagnostopoulos et al. (2006) is the *SwapRounds* operator and swaps the matches of two rounds, r_x and r_y. The move operator $SwapRounds(S, r_x, r_y)$ will swap the matches of r_x and r_y for every team that participates in the competition. An example of this operator is provided in Table 7.

T/R	1	2	3	4	5	6
1	4	3	-2	-3	2	-4
2	-3	-4	1	4	-1	3
3	2	-1	-4	1	4	-2
4	-1	2	3	-2	-3	1

T/R	1	2	3	4	5	6
1	4	-3	-2	3	2	-4
2	-3	4	1	-4	-1	3
3	2	1	-4	-1	4	-2
4	-1	-2	3	2	-3	1

Table 7: Example $SwapRounds(S, r_2, r_4)$

3.3.1.3 Swap Teams

The move operator *SwapTeams* proposed by Anagnostopoulos et al. (2006) swaps the schedule of two teams, t_i and t_j, except when they play each other. This implies that two matches will change in $2(n-2)$ rounds. This will result in a new schedule for n teams where $4(n-2)$ matches have been changed. Table 8 provides an example of this move operator.

T/R	1	2	3	4	5	6
1	4	3	-2	-3	2	-4
2	-3	-4	1	4	-1	3
3	2	-1	-4	1	4	-2
4	-1	2	3	-2	-3	1

T/R	1	2	3	4	5	6
1	-2	3	-4	-3	4	2
2	1	-4	3	4	-3	-1
3	4	-1	-2	1	2	-4
4	-3	2	1	-2	-1	3

Table 8: Example $SwapTeams(S, t_1, t_3)$

3.3.1.4 Partial Swap Teams

The move operator *PartialSwapTeams* of Anagnostopoulos et al. (2006) involves one round, r_x, and two teams, t_i and t_j. A precondition for this move is that t_i and t_j do not play each other in round r_x but have 2 different opponents, e.g. t_a and t_bn respectively. It does not matter if they play at home or away. The move will start by swapping both matches, $t_i - t_a$ and $t_j - t_b$ to $t_i - t_b$ and $t_j - t_a$. After the swap, a schedule will appear that violates some

hard constraints (C1 and C2 (Section 2.1)) of a double round-robin tournament. To restore those violations, an ejection chain is called upon. This chain starts by searching the round r_a where the original match $t_i - t_b$ was planned. In r_a, the matchups of t_i and t_j are swapped again. Matches $t_c - t_i$ and $t_d - t_j$ will now become $t_d - t_i$ and $t_c - t_j$. After this swap, the ejection chain searches again for the round where the match $t_d - t_i$ was originally planned and swaps the matchups for both t_i and t_j. This procedure is repeated until the circle is complete, i.e. we arrive again in round r_x. After the ejection chain is applied, a schedule will be produced that is part of the search space (but not necessarily of the solution space). The maximum number of rounds where matches can be swapped is equal to $|R| - 2$, because the rounds where t_i and t_j have a match-up will never change. For every involved round, two matches will be altered.

The example of this move operator in Table 9 requires some additional clarification. After matches $t_4 - t_2$ and $t_3 - t_6$ are changed to $t_6 - t_2$ and $t_3 - t_4$ in r_5 a schedule appears that is not a part of the search space. An ejection chain is invoked to make sure that the new schedule is again part of the search space. First, the round of the initial match $t_3 - t_4$ is determined, here r_1. In r_1, the original matches $t_3 - t_4$ and $t_1 - t_6$ are replaced by $t_3 - t_6$ and $t_1 - t_4$. Next, the round of initial match $t_1 - t_4$ is determined, here r_6. Also for this round the original matches $t_1 - t_4$ and $t_5 - t_6$ are replaced by $t_1 - t_6$ and $t_5 - t_4$. This procedure repeats itself for r_2, r_7, r_8, r_4 and r_9, in that particular order. The matches in r_9 were originally $t_4 - t_5$ and $t_6 - t_2$ and are replaced by $t_6 - t_5$ and $t_4 - t_2$. The match $t_4 - t_2$ was originally planned to be played in r_5, the initial round. After this last swap, the circle is complete and the schedule is again part of the search space. However, the resulting schedule is not part of the solution space because the atmost constraint is violated by team 4 in round 10. A fourth consecutive home game exceeds the upper bound of 3 set on home stands or away trips. Since this new schedule is infeasible, additional moves will have to be applied to retrieve a feasible schedule.

T/R	1	2	3	4	5	6	7	8	9	10
1	6	3	-2	-4	5	4	-5	-6	-3	2
2	-5	6	1	3	4	-3	4	5	-6	-1
3	4	-1	5	-2	6	2	-6	-4	1	-5
4	-3	-5	-6	1	2	-1	-2	3	5	6
5	2	4	-3	-6	-1	6	1	-2	-4	3
6	-1	-2	4	5	-3	-5	3	1	2	-4

T/R	1	2	3	4	5	6	7	8	9	10
1	4	3	-2	-6	5	6	-5	-4	-3	2
2	-5	4	1	3	-6	-3	6	5	-4	-1
3	6	-1	5	-2	4	2	-4	-6	1	-5
4	-1	-2	-6	5	-3	-5	**3**	**1**	**2**	**6**
5	2	6	-3	-4	-1	4	1	-2	-6	3
6	-3	-5	4	1	2	-1	-2	3	5	-4

Table 9: Example $PartialSwapTeams(S, t_4, t_6, r_5)$

3.3.1.5 Partial Swap Rounds

The last move operator proposed by Anagnostopoulos et al. (2006) is called *PartialSwapRounds* and is determined by one team t_i and two rounds, r_x and r_y. The move will swap the opponents of team t_i in rounds r_x and r_y. This swap will create a schedule that falls outside the search space, which calls for an ejection chain to restore the violations on C1 and C2 (Section 2.1). It is sufficient to find the connected component which contains the games of t_i, in r_x and r_y. A connected component is a graph where the nodes represent the teams and where an edge connects two teams that play against each other in r_x or r_y (see e.g. Figure 2). All teams in this connected component must have their games swapped (Anagnostopoulos et al., 2006). The opponents on the other hand might have to change their location to meet the requirements for the new match-up. This will create a new schedule that falls within the search space. The matches of only two rounds will be altered for this move operator.

The example in Table 10 again requires some additional clarification. Swapping the games of t_1 in r_6 and r_{10} will lead to a schedule that does not meet the requirements of a double round-robin tournament. The connected component that contains the games of t_1 in r_6 and r_{10} ($t_1 - t_4$,$t_1 - t_2$) also includes the arcs $t_4 - t_6$,$t_5 - t_6$,$t_5 - t_3$,$t_3 - t_2$ (Figure 2). Since it is necessary to swap the games in r_6 and r_{10} of all other nodes in the connected component, the matches of the other teams involved: t_2, t_3, t_4, t_5 and t_6 in r_6 and r_{10} should be swapped. The resulting schedule is part of both the search and solution space. However, the latter is not guaranteed by this operator.

T/R	1	2	3	4	5	6	7	8	9	10
1	6	3	-2	-4	5	4	-5	-6	-3	2
2	-5	6	1	3	-4	-3	4	5	-6	-1
3	4	-1	5	-2	6	2	-6	-4	1	-5
4	-3	-5	-6	1	2	-1	-2	3	5	6
5	2	4	-3	-6	-1	6	1	-2	-4	3
6	-1	-2	4	5	-3	-5	3	1	2	-4

T/R	1	2	3	4	5	6	7	8	9	10
1	6	3	-2	-4	5	2	-5	-6	-3	4
2	-5	6	1	3	-4	-1	4	5	-6	-3
3	4	-1	5	-2	6	-5	-6	-4	1	2
4	-3	-5	-6	1	2	6	-2	3	5	-1
5	2	4	-3	-6	-1	3	1	-2	-4	6
6	-1	-2	4	5	-3	-4	3	1	2	-5

Table 10: Example $PartialSwapRounds(S, t_1, r_6, r_{10})$

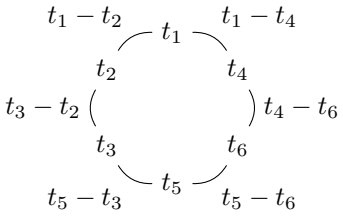


Figure 2: Connected Component $PartialSwapRounds(S, t_1, r_6, r_{10})$

3.3.2 Algorithms and Heuristics

In this section three relevant algorithms for TTP are discussed. The tabu search approach (Di Gaspero & Schaerf, 2007) is covered since it introduces the ejection chain as it will be used in this dissertation. The research of Di Gaspero and Schaerf was also used to broaden the knowledge on the move operators proposed in Anagnostopoulos et al. (2006). Secondly, simulated annealing heuristics are covered. The PBSA algorithm by Van Hentenryck and Vergados (2007) delivered the current best known solutions for the NLx-instances and TTSA, proposed by Anagnostopoulos et al. (2006), serves as the underlying mechanism for it. These algorithms are therefore relevant to cover in this section.

3.3.2.1 Tabu Search Heuristics

A first algorithm used in research on the TTP is a tabu search based heuristic. The tabu search approach for TTP was introduced by Di Gaspero and Schaerf (2007). This method dives deeper into the neighbourhood of certain moves and makes use of a tabu list in order to search for new schedules in a more efficient way. The program is extended with a local search algorithm to improve the search for a (near) optimal solution. The tabu search heuristic is equipped with its own cost function based on the total distance to be travelled and the number of violated constraints. Two constraints, C4 and C5 (Section 2.1), are defined as soft constraints. Violations against these create an infeasible schedule and are penalised in the cost function. To perform a local search, the algorithm uses the five move operators proposed by Anagnostopoulos et al. (2006):

1. $SwapHomes$: swap venues for two teams in two different rounds
2. $SwapTeams$: swap two teams in all rounds
3. $SwapRounds$: swap two complete rounds
4. $PartialSwapTeams$: change the opponents for two teams in one round
5. $PartialSwapRounds$: swap the opponents of one team in two rounds

Since the last two move operators might create a schedule that falls outside the search space, an ejection chain should be included (Di Gaspero & Schaerf, 2007). An ejection chain for the TSP is defined by Glover (1992) as follows: *"An ejection chain is initiated by selecting a set of elements to undergo a change of state (e.g. to occupy new positions or receive new values). The result of this change leads to identifying a collection of other sets, with the property that the elements of at least one must be "ejected from" their current states."* The translation to the TTP problem is the following. When a schedule undergoes a certain swap operation where several elements are changed, the newly created schedule might fall outside the search space, i.e. violations against constraints C1, C2 or C3 (Section 2.1) can occur. This calls for an ejection chain that will change critical elements in the schedule in order to make it part of the search space again. However, it is not guaranteed that the new schedule will be part of the solution space, i.e. the new schedule might violate constraints C4 and C5.

To improve efficiency and to avoid not detecting a move on the tabu list, Di Gaspero and Schaerf (2007) execute a neighbourhood analysis to determine

move operators that have the same end result. The cardinality of both move operator 3 & 5, and 4 & 5 are significant i.e. the execution of both move operators will often result in the same schedule. Finally the quality of the move operators is observed. The quality is defined as the variation in cost due to the move. The conclusion is that the best moves in terms of quality are move 1, move 4 with a move length of 4 or smaller and move 5 with a move length of 6 or smaller. For move 4 the shortest chain closure (SCC) , which consists in closing the chain as soon as one of the two closing matches is encountered, should be used to obtain optimal results. These 3 moves will lead to better schedules in a more efficient way. Using this algorithm and especially these 3 moves, the best solutions of certain problems can be approached. The tabu search approach is proven to be a decent method to solve the TTP.

3.3.2.2 Simulated Annealing Heuristics

Simulated annealing heuristics are based on the annealing process found in metallurgy. Oxford Learner's Dictionary defines this process as '*heating metal or glass and allowing it to cool slowly, in order to make it stronger or softer*'[8]. Kirkpatrick, Gelatt, and Vecchi (1983) simulated this process to be used as an optimisation algorithm. The first step consists of 'melting' the system to be optimised at a high temperature. Then, the temperature is slowly lowered until the system 'freezes' and no more changes occur. The temperature should only adapt when the systems reaches a steady state. One important feature of simulated annealing heuristics is that local optima can always be escaped at nonzero temperatures (Kirkpatrick et al., 1983). In the following sections two relevant simulated algorithms are discussed.

3.3.2.2.1 TTSA

A first simulated annealing heuristic is proposed by Anagnostopoulos et al. (2006). This algorithm improved the solutions for the NL12 , NL14 and NL16 instances. Based on existing simulated annealing techniques, the travelling tournament simulated annealing (TTSA) algorithm was developed. Some main design characteristics define this new method and lead to high-quality solutions.

The first characteristic is the separation of constraints into two groups: hard constraints, which are always satisfied by the configurations, and soft constraints, which can be violated. A schedule that violates one or more soft constraints is called an infeasible schedule. There are basically two soft constraints for the TTP. The 'no repeat' constraint states that teams should not play two consecutive matches against the same opponent. The 'atmost' constraint limits the number of consecutive home or away games (Van Hentenryck & Vergados, 2007). Due to the implementation of soft constraints, TTSA can explore both the feasible and the infeasible region which seems particularly important for this problem (Anagnostopoulos et al., 2006).

The second characteristic arises from the selection of the move operators. The set of move operators ensures that the neighbourhood of any schedule is as large as possible. This large neighbourhood provides an extensive number of possible schedules that can be obtained from any given schedule. Attention

[8] `https://www.oxfordlearnersdictionaries.com/definition/american_english/anneal`

should be paid to move operators that only perform a swap between two teams. These operators may require the change of $4(n-2)$ entries in a schedule to make it part of the search space again. A similar concept of ejection chains is also used in the tabu search approach of Di Gaspero and Schaerf (2007) (cf. Section 3.3.2.1).

A third characteristic of TTSA is the implementation of a strategic oscillation strategy. By dynamically adjusting the objective function, the time spent in the feasible and infeasible region is balanced. This is done by adjusting the weight factor w, which penalises each violation of soft constraints with a specific weight, whenever a new schedule is found. After spending a long time in the infeasible region, w will become large and the penalty for violations increases with it. This will drive the search towards the feasible region.

A last design characteristic is the usage of reheats in order to facilitate the escape from local minima in search of the global minimum. The exploration of the neighbourhood is done by using the 5 basic move operators discussed in Section 3.3.1.

Since this dissertation will start from the TTSA, the next paragraphs contain an extensive overview of its structure. The algorithm starts by generating a random initial feasible schedule S. The objective function value of this initial schedule can be calculated according to the following rules. If the schedule is feasible, the objective function will be the total cost (here the total distance) of the schedule. If the schedule is infeasible, the total cost is incremented with a penalty factor.

The objective function of a schedule S is given by the following formula (with $nbv(S)$ being the number of violations against the soft constraints of schedule S)

$$C(S) = \begin{cases} totalDistance(S) & \text{if } S \text{ is feasible} \\ \sqrt{totalDistance(S)^2 + [w * f(nbv(S))]^2} & , \quad \text{otherwise} \end{cases}$$

The rationale behind the definition of f is an interesting one to include. It is intuitively clear that having 1 violation instead of 0, i.e. crossing the feasible/infeasible boundary, should cost more than any subsequent violations. Adding 1 violation to a schedule that already has 10 violations does not make that much difference. In their experiments Anagnostopoulos et al. have chosen $f(nbv(S)) = 1 + (\sqrt{nbv(S)} * \ln(nbv(S)))/2$. The size of the penalty factor depends on the value of the weight factor w. When a better feasible schedule is found, w is divided by a factor θ ($\theta > 1$). If a better infeasible schedule is found, w is multiplied by θ. As a result, w will vary according to the frequencies of feasible and infeasible configurations in the last iterations, this is the implementation of the strategic oscillation strategy. Strategic oscillation will balance the time spent in feasible and infeasible regions of the search space.

After the initialisation, a series of iterations will be performed on the schedule. The algorithm randomly applies one of the move operators and calculates the objective function value for this new schedule. Each iteration ends with calculating the variation Δ which is the difference between the objective function values of the newly created schedule and the best-known schedule in the algorithm. If $\Delta \geq 0$, TTSA applies the move operator and

changes the best schedule to the new schedule. Otherwise, it applies the move with a certain probability equal to $\exp(-\Delta/T)$. T is used as the temperature in this algorithm and varies continuously during the execution of the program. The temperature decreases each iteration since it is multiplied with a factor β ($\beta \in [0, 1]$), the cooling factor. Temperature T can also be increased to facilitate moving away from a local minimum. This is the reheating procedure, the last of the design characteristics mentioned in previous paragraphs. After a certain amount of iterations in which no better schedule was found, a reheat increases the temperature to twice the value of T when the last schedule that improved the objective function value was found. The algorithm ends when the upper limit of non-improving iterations is met.

3.3.2.2.2 PBSA

A second simulated annealing-based algorithm, population-based simulated annealing (PBSA), is actually an extension of the TTSA algorithm. Van Hentenryck and Vergados (2007) extended the TTSA method and found new best solutions for NL12, NL14 and NL16 and other large-scale instances. The simulated annealing approach is used because of its great capabilities in micro-intensification and -diversification of schedules. To improve this algorithm, macro-intensification and -diversification should be added. This is typically found in population-based and tabu search algorithms. Therefore, TTSA is used as a basis and some components are added to create the population-based simulated annealing algorithm.

Basically, the core of the algorithm is organised as a series of waves. The first wave consists of x times (e.g. 25 times) the same initial schedule on which a simulated annealing run is performed. This will result in x new best schedules. From those x new best schedules, the best y ($y \leq x$) (e.g. 5) are called the elite runs and will be kept for the next wave. The other $x - y$ (e.g. $25 - 5 = 20$) will start from the schedule with the current best objective function value. This procedure repeats itself with each wave having its own set of starting schedules and temperature.

The core of the algorithm terminates when a certain number of waves does not adapt the best schedule. This core procedure will be restarted on a lower temperature until a new schedule is found. The algorithm is stopped when no new best schedule is found for a certain amount of core procedures. Van Hentenryck and Vergados have also investigated the effect of this macro-diversification (adding the population-based layer). They concluded among other things that an increasing number of elite runs will improve the solutions but will also increase the time consumed to run the algorithm.

3.3.3 Construction of a TTP Schedule

The discussed algorithms all require the initialisation or the construction of a schedule. The move operators will then be performed on this initial schedule. Trick (2000) proposes a multiphase approach for the construction of sports timetables and defines three separate subproblems. Finding home-away patterns (HAPs) for all n teams in a competition can be considered as a first subproblem. This consists of finding a set of n strings of pluses and minuses that correspond to the home and away sequence of a team. Trick (2000) defines the construction of a basic match schedule (BMS) as a second

subproblem. Constructing this BMS consists out of assigning games to the HAPs that where proposed in the first subproblem. The basic match schedule is actually a schedule with generic teams. The third subproblem consists of changing these generic teams into real teams, which then completes the timetable.

The alternative approaches in sports timetable generation mainly differ in the order in which these subproblems are being tackled. In general, the more critical a decision is perceived the earlier it will be made. The second and third subproblem can also be considered as one by immediately using real teams in developing a schedule. The order of the two remaining subproblems is dependent on the importance the researchers assign to the individual subproblems. Russell and Leung (1994) propose a first-break-then-schedule approach, i.e. they first generate home-away patterns and then they assign teams to these patterns. Trick (2000) proposes a first-schedule-then-break approach, which reverses the order of these subproblems. A first phase fixes the opponents for all rounds. The resulting schedule can be called the *opponent schedule*, in which each team is represented by a different integer. In a second phase, the decision about the venue of every match in every round is made. This will result in the home-away assignment for the opponent schedule. This assignment respects the opponent schedule and decides for every game which of the two competing teams will play at home (+) and which team will play an away game (-). This results in a complete schedule of a round-robin tournament.

The second phase of the first-schedule-then-break approach can also be seen as a new optimisation problem. Rasmussen and Trick (2006) define this problem as the timetable constrained distance minimisation problem (TCDMP). For a given timetable of a double round-robin tournament with n teams, an $n \times n$ distance matrix and an upper bound on the amount of consecutive home and consecutive away games, this problem determines an optimal home-away assignment. Rasmussen and Trick (2006) provide both an IP- and a CP-formulation for this problem. Apart from these formulations, the authors also present two alternative solution methods for the TCDMP: a hybrid integer programming/constraint programming approach and a branch and price algorithm. However, the IP-formulation, and to a lesser extent the CP-formulation, are the most relevant to cover in this master's dissertation. For the exact models and more details on the formulations, the reader is referred to Rasmussen and Trick (2006). Since the TCDMP only covers the home-away assignment of given timetable, several constraints present in the TTP can be omitted, e.g. the home-away assignment has no impact on the violation of the no-repeat constraint. Therefore, this problem can be seen as less challenging than the TTP.

In the results of their research, Rasmussen and Trick (2006) show that the IP-formulation is better equipped to handle larger instances than the CP-formulation. This is not seen as a surprise since IP-models frequently excel for optimisation problems, when they are compared to CP models. However, the CP-formulation is shown to be better at solving instances with 6 teams. Additionally, the simplicity of CP-models as opposed to IP-models, makes it clear that both types have their advantages.

Since using a high-quality approximation of an optimal solution as initial schedule can significantly improve the results of any given algorithm, the 5.875 approximation algorithm by Westphal and Noparlik (2014) is also covered

here. The schedule resulting from this algorithm approximates the optimal TTP solution by a factor which is not more 5.875 for any $U \leq 4$ and any $n \leq 6$. It also produces this high quality solutions in a very short timeframe.

First, the teams are ordered by descending star weight, i.e. the team with the largest total distance to all other teams gets assigned to the first place. For clarity's sake, in Figure 3 all teams are represented by their place in this ordered list, i.e. the team represented by number 1 is the team with the highest star weight. For n=20, the first and second round are displayed in Figure 3.

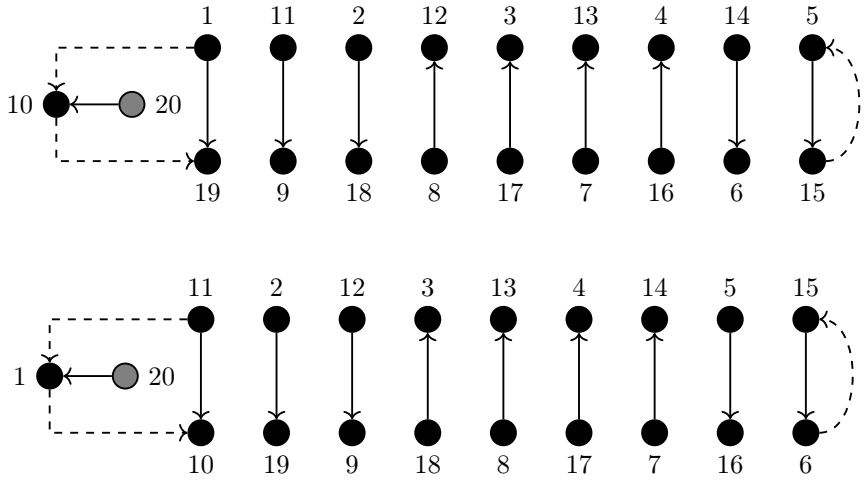


Figure 3: Construction Round 1 and 2 ($n = 20, U = 4$)

The graph consists out of nodes and oriented arcs. A solid arc (u, v) means that team u plays against team v at the venue of team v. Games for the following rounds of the first half of the season can be derived by rotating the positions of the teams counterclockwise. Only one arc changes orientation during one half of the season, this is the arc that is connected with the node with the smallest star weight, in this case node 20. This arc changes orientation every Uth round. The predetermined orientation of the arcs makes sure that no team has home stands or away trips with a size exceeding U. By rotating the positions, every team will have met every other team exactly once during this first $n-1$ rounds.

To construct a complete schedule of a double round-robin tournament, the second half of the season remains to be constructed. To avoid home stands and away trips that exceed the allowable length, Westphal and Noparlik (2014) let the second half of the season start with the matches scheduled in round $n-2$, followed by the matches of round $n-1, 1, 2, ..., n-3$. Of course the venues for the second half of the competition are swapped. For an even more detailed overview of this algorithm and the rationale behind it, the reader is referred to Westphal and Noparlik (2014).

3.4 RTTP

While the previous section mainly focused on research in the time-constrained environment, this section covers previous research on the time-relaxed travelling tournament problem. First some move operators that were used in previous research on the RTTP are covered. This is followed by a description of the two algorithms that have provided the best known solutions for the NLx instances.

3.4.1 Basic Move Operators from TTP

A first set of important move operators are those presented by Anagnostopoulos et al. (2006). However, these moves are developed for a time-constrained environment and some of them are not able to cope with the presence of byes. These moves were adapted to a time-relaxed context by Pérez-Cáceres and Riff (2015).

3.4.1.1 Swap Homes RTTP

The $SwapHomesRTTP$ move operator does not differ from its time-constrained counterpart.

T/R	1	2	3	4	5	6	7
1	4	3	-2	-3	0	2	-4
2	3	-4	1	4	0	-1	-3
3	-2	-1	-4	1	4	0	2
4	-1	2	3	-2	-3	0	1

T/R	1	2	3	4	5	6	7
1	4	3	-2	-3	0	2	-4
2	-3	-4	1	4	0	-1	3
3	2	-1	-4	1	4	0	-2
4	-1	2	3	-2	-3	0	1

Table 11: Example $SwapHomesRTTP(S, t_2, t_3)$

3.4.1.2 Swap Rounds RTTP

The $SwapRoundsRTTP$ move operator also remains unchanged after conversion from TTP to RTTP.

T/R	1	2	3	4	5	6	7
1	4	3	-2	-3	0	2	-4
2	3	-4	1	4	0	-1	-3
3	-2	-1	-4	1	4	0	2
4	-1	2	3	-2	-3	0	1

T/R	1	2	3	4	5	6	7
1	4	-3	-2	3	0	2	-4
2	-3	4	1	-4	0	-1	3
3	2	1	-4	-1	4	0	-2
4	-1	-2	3	2	-3	0	1

Table 12: Example $SwapRoundsRTTP(S, r_2, r_4)$

3.4.1.3 Swap Teams RTTP

The conversion to a time-relaxed environment changes the number of matches that are altered. The time-constrained version of the *SwapTeams* operator swaps 2 matches, the matches of t_i and t_j in $2(n-2)$ rounds (all rounds where t_i and t_j do not have a matchup). The resulting schedule will have $4(n-2)$ different matches compared to the original schedule. The RTTP variant has an additional k rounds, meaning an additional $2k$ matches have to be altered. However, if both teams have a bye in the same round, no matches have to be changed for that round. The parameter s indicates the number of times t_i and t_j have a bye in the same round. This move operator will result in $2(n-2)+2(k-s)$ different matches as compared to the original schedule in a competition with n teams and k byes.

T/R	1	2	3	4	5	6	7
1	4	3	-2	-3	0	2	-4
2	-3	-4	1	4	0	-1	3
3	2	-1	-4	1	4	0	-2
4	-1	2	3	-2	-3	0	1

T/R	1	2	3	4	5	6	7
1	-2	3	-4	-3	4	0	2
2	1	-4	3	4	0	-3	-1
3	4	-1	-2	1	0	2	-4
4	-3	2	1	-2	-1	0	3

Table 13: Example $SwapTeamsRTTP(S, t_1, t_3)$

3.4.1.4 Partial Swap Teams RTTP

The move $PartialSwapTeamsRTTP$ is performed in a similar way as its time-constrained counterpart. The move swaps the opponents of two teams, t_i and t_j in round r_a. A precondition for this move is that t_i and t_j do not play each other in round r_a but have 2 different opponents, t_a and t_b. It is possible that one of both teams has a bye in r_a. However, it is not allowed for both t_i and t_j to have a bye in r_a since this would cause the schedule to remain unchanged. The move will start by swapping matches $t_i - t_a$ and $t_j - t_b$ to $t_i - t_b$ and $t_j - t_a$. After this swap, a schedule will appear that violates at least some of the hard constraints (C1, C2 and C3) of a time-relaxed double round-robin tournament. To restore those violations, an ejection chain that works almost exactly as in Section 3.3.1.4, is called upon. The only difference occurs when one of the teams, e.g. t_i, has a bye in a particular round, e.g. r_b. If this is the case, t_j will become the team that has a bye in r_b and t_i will now have a match-up with the former opponent of t_j in r_b. To continue the ejection chain, a round where t_j originally had a bye has to be found. When multiple byes occur for a single team, a list of all visited and swapped rounds is stored and updated to make sure no round is visited twice. After a round is selected where t_j previously had a bye, the ejection chain can continue until the circle is complete and round r_a is selected again. After the ejection chain is applied, the returned schedule is part of the search space but not necessarily of the solution space. The maximum number of rounds where matches can be swapped is equal to $|R| - 2 - s$ (with s = number of rounds where both t_i and t_j have a bye).

T/R	1	2	3	4	5	6	7	8	9	10	11
1	4	0	-2	-6	5	3	-4	2	-3	6	-5
2	3	-5	1	4	-6	0	5	-1	-4	-3	6
3	-2	0	-6	5	4	-1	6	-5	1	2	-4
4	-1	6	5	-2	-3	0	1	-6	2	-5	3
5	-6	2	-4	-3	-1	6	-2	3	0	4	1
6	5	-4	3	1	2	-5	-3	4	0	-1	-2

T/R	1	2	3	4	5	6	7	8	9	10	11
1	4	0	-2	-6	3	5	-4	2	-5	6	-3
2	5	-3	1	4	-6	0	3	-1	-4	-5	6
3	-6	2	-4	5	-1	6	-2	-5	0	4	1
4	-1	6	3	-2	-5	0	1	-6	2	-3	5
5	-2	0	-6	-3	4	-1	**6**	**3**	**1**	**2**	-4
6	3	-4	5	1	2	-3	-5	4	0	-1	-2

Table 14: Example $PartialSwapTeams(S, t_3, t_5, r_{10})$

In this paragraph, the example in Table 14 is explained in more detail. After matches $t_3 - t_2$ and $t_5 - t_4$ are swapped to $t_3 - t_4$ and $t_5 - t_2$ in r_5 a schedule appears that does not comply with the requirements of a double round-robin tournament. The ejection chain will restore this as follows. First, the round where $t_3 - t_4$ was originally scheduled is obtained, yielding r_5 for our example. In r_5, the original matches $t_3 - t_4$ and $t_1 - t_5$ are replaced by $t_1 - t_3$ and $t_5 - t_4$. This procedure repeats itself for $r_6, r_7, r_1, r_3, r_{11}$ and r_9, in that particular order. The matches in r_9 were originally $t_3 - t_1$ and $t_5 - bye$ and are now replaced by $t_3 - bye$ and $t_5 - t_1$. The bye of t_3 was originally planned in r_2, a round that has not been visited before. For r_2, the original matches $t_3 - bye$ and $t_5 - 2$ and are now replaced by $t_3 - t_2$ and $t_5 - bye$. The match $t_3 - t_2$ was originally planned to be played in r_{10}, which was the initial round. After this last swap, the circle is complete and the schedule is again part of the search space. However, the resulting schedule is not part of the solution space because the atmost constraint is violated by team 5 in round 10. A fourth consecutive home game exceeds the upper bound of three games set on home stands or away trips.

3.4.1.5 Partial Swap Rounds RTTP

The last move operator, $PartialSwapRoundsRTTP$, swaps the opponents of a team t_i in round r_x and round r_y, just like its time-constrained counterpart (Section 3.3.1.5). After swapping the opponents in r_x and r_y, an ejection chain is called upon to restore the violations of C1, C2 and C3 (Section 2.1). The connected component of t in r_x and r_y has to be found and all the teams in this component must have their games swapped. It is possible that t_i has a bye in r_x or r_y. If t_i has a bye in both r_x and r_y, the resulting schedule will not differ from the original schedule since the connected component will only consist of $t_i - 0$ and $t_i - 0$. Therefore this situation should be avoided. This move will only change elements of the two rounds r_x and r_y.

T/R	1	2	3	4	5	6	7	8	9	10	11
1	4	0	-2	-6	5	3	-4	2	-3	6	-5
2	3	-5	1	4	-6	0	5	-1	-4	-3	6
3	-2	0	-6	5	4	-1	6	-5	1	2	-4
4	-1	6	5	-2	-3	0	1	-6	2	-5	3
5	-6	2	-4	-3	-1	6	-2	3	0	4	1
6	5	-4	3	1	2	-5	-3	4	0	-1	-2

T/R	1	2	3	4	5	6	7	8	9	10	11
1	-3	0	-2	-6	5	3	-4	2	4	6	-5
2	-4	-5	1	4	-6	0	5	-1	**3**	**-3**	6
3	1	0	-6	5	4	-1	6	-5	**-2**	**2**	-4
4	2	6	5	-2	-3	0	1	-6	-1	-5	3
5	-6	2	-4	-3	-1	6	-2	3	0	4	1
6	5	-4	3	1	2	-5	-3	4	0	-1	-2

Table 15: Example $PartialSwapRoundsRTTP(S, t_1, r_1, r_9)$

Swapping the games of t_1 in r_1 and r_9 will lead to a schedule that does not meet the requirements of a double round-robin tournament. The connected component that contains the games of t_1 in r_1 and r_9 ($t_1 - t_4$,$t_1 - t_3$) also includes the arcs $t_4 - t_2$,$t_2 - t_3$ (Figure 4). It is necessary to swap the games in r_1 and r_9 of all other nodes in this connected component, to be more precise the matches of the other teams involved, t_2, t_3 and t_4, should be swapped. The result is a schedule that is part of the search space but not of the solution space. The latter because the new schedule in infeasible since it violates the no repeat constraint twice (in round 10 by team 2 and 3).

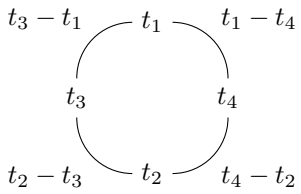


Figure 4: Connected Component $PartialSwapRoundsRTTP(S, t_1, r_1, r_9)$

3.4.2 Additional Move Operators

In addition to the adaptation of these five move operators (Anagnostopoulos et al., 2006), Pérez-Cáceres and Riff (2015) proposed two new operators, that are only applicable in a time-relaxed environment since they exploit the presence of byes. Similar to the definition in this dissertation, Pérez-Cáceres and Riff define an away pattern as a sequence of away games and potentially byes, without any home game in between. The concept of an 'extendable away group' was defined as *"an away pattern that can be modified in order to increase the continuous sequence size of non-character +"*." Basically, any away trip with less than three consecutive away games is an extendable away group since an extra away game can be added to the group. When working with an RTTP with 1 bye per

team and an upper limit on the length of away trips of 3, the set of extendable away groups is $\{-, 0-, -0, 0-0, --, 0--, --0, -0-\}$ (with '$-$' representing an away game and '0' representing a bye). A final move operator, $ShiftMove$, was proposed by Chen, Kendall, and Vanden Berghe (2007).

3.4.2.1 Swap Bye Matches

The move operator $SwapByeMatches$ proposed by Pérez-Cáceres and Riff (2015) is only applicable for time-relaxed problems. First two teams, t_i and t_j, that have a bye in the same round are selected. The operator consists out of swapping the byes for both teams with a round where they have a match-up. This means that one of their byes originally was scheduled in round r_x and one of their match-ups in round r_y (the choice for r_y will be made randomly). This will be changed to a bye in r_y and a match-up in r_x. This easy to implement move operator can potentially extend an away pattern of a team. By extending an away pattern of a team from 1 to 2 or from 2 to 3, more consecutive away games will occur and this can reduce the total distance to be travelled.

T/R	1	2	3	4	5	6	7
1	4	-3	-2	0	2	-4	3
2	3	-4	1	4	-1	-3	0
3	-2	1	-4	0	4	2	-1
4	-1	2	3	-2	-3	1	0

T/R	1	2	3	4	5	6	7
1	4	-3	-2	0	2	-4	3
2	3	0	1	4	-1	-3	-4
3	-2	1	-4	0	4	2	-1
4	-1	0	3	-2	-3	1	2

Table 16: Example $SwapByeMatches(S, t_2, t_4)$

3.4.2.2 Grouping Away Using Byes

This move operator focuses on extending an away trip of a team in order to reduce its travelling distance. First a random extendable away group is selected. For team t, where this away group belongs to, a round r_x, where team t has an away game (preferably one that is not part of an away trip) and that is not covered by the extendable away group, is randomly selected. The next step consists out of performing $SwapRoundsRTTP(S, r_x, r_y)$, with r_y the round preceding or following the extendable away group. When the extendable away group is both preceded and followed by a round, one of the two is randomly chosen as round r_y.

T/R	1	2	3	4	5	6	7
1	4	0	-2	-3	2	-4	3
2	3	-4	1	4	-1	-3	0
3	-2	0	-4	1	4	2	-1
4	-1	2	3	-2	-3	1	0

T/R	1	2	3	4	5	6	7
1	4	3	-2	-3	2	-4	0
2	3	0	1	4	-1	-3	-4
3	-2	-1	-4	1	4	2	0
4	-1	0	3	-2	-3	1	2

Table 17: Example $GroupingAwayUsingByes(S)$

3.4.2.3 Shift Move

The move operator $ShiftMove$ was proposed by Chen et al. (2007) and works as follows. Randomly select rounds r_x and r_y ($x < y$), remove r_x and insert it at the position of round y, round r_y and all rounds in between will now be scheduled one round earlier.

T/R	1	2	3	4	5	6	7
1	4	0	-2	-3	2	-4	3
2	3	-4	1	4	-1	-3	0
3	-2	0	-4	1	4	2	-1
4	-1	2	3	-2	-3	1	0

T/R	1	2	3	4	5	6	7
1	4	-2	-3	2	0	-4	3
2	3	1	4	-1	-4	-3	0
3	-2	-4	1	4	0	2	-1
4	-1	3	2	-3	2	1	0

Table 18: Example $ShiftMove(S, r_2, r_5)$

3.4.3 Algorithms and Heuristics

In this section the most important algorithms and heuristics for the RTTP are covered. Both the complete search method by Brandão and Pedroso (2014) and the clonal selection algorithm by Pérez-Cáceres and Riff (2015) came up with a series of new best solutions for an extensive amount of instances and are therefore covered here.

3.4.3.1 Complete Search Method

A first RTTP algorithm, the Complete Search Method, was created by Brandão and Pedroso (2014). This technique can be seen as the first heuristic approach that can present solutions for an extensive amount of RTTP instances. Brandão and Pedroso abandoned the brute force-method and came up with an algorithmic approach for the problem. Their method contains several tools: branch-and-bound, metaheuristics and dynamic programming. The branch and bound method is used to generate solutions round by round.

Metaheuristics are added to the algorithm to improve the bounds of the solution. Dynamic programming on its turn is used to compute the independent lower bounds quickly and efficiently after each iteration. The operators used by the method are $SwapHomesRTTP, SwapRoundsRTTP$ and $SwapTeamsRTTP$ (Section 3.4.1).

Generating schedules is done round by round, i.e. all games of round r_x must be fixed before moving on to round r_{x+1}. By doing this, restrictions can be detected much earlier and excess calculations are avoided. A branch will not be further investigated if the sum of the current cost and the independent lower bound of a candidate is higher than the upper bound. This independent lower bound of all teams is the sum of the optimal schedule for every team, ignoring any restrictions or validity constraints (Easton, Nemhauser, & Trick, 2002). In order to calculate this independent lower bound, a dynamic programming approach is used to determine the optimal distance for each team given their location and the amount of matches they have already played.

Finally, hill-climbing metaheuristics are used to improve bounds. When a local minimum is reached, random perturbations are applied until a new minimum is reached or until a number of iterations is performed. Each iteration consists of applying one of the move operators to the schedule and checking if this generates a new better schedule. The three transformations (or operators) do not cover the complete search space and as a consequence lead to suboptimal solutions. In addition, some auxiliary data structures are stored to check several constraints in a fast way. These data structures consist of the current location, last opponent, number of byes used, number of consecutive home or away games and a matrix to check if a game has already been scheduled. Using the complete search method, competitive solutions are found for NL4, NL6 and NL8 with up to 3 byes. A noteworthy finding is the longer runtime of the RTTP in comparison with the TTP, supporting the claim from Section 2.4.1 that the RTTP at least matches the complexity of the TTP.

3.4.3.2 Clonal Selection Algorithm

A second proposed algorithm, 'CLONALG', is created by Pérez-Cáceres and Riff (2015) and has generated most of the best-known solutions for the time-relaxed NLx instances [9,10]. Pérez-Cáceres and Riff (2015) introduce new components to the Clonal Selection Algorithm (CLONALG), an artificial immune algorithm based on clonal selection principle. The new version of CLONALG is called $RAIS_{TTP}$ (artificial immune system for relaxed TTP). It can handle hard constraints that define the search space and soft constraints. The algorithm uses a penalty and fitness function to find solutions that satisfy the constraints. The penalty function consists of the average distance among teams and a penalty factor associated to the violation of a constraint. The fitness function is defined by the total travelled distance of all teams and the number of violated atmost and no repeat constraints in scheduling candidates. To make changes to the candidate solution, the five basic transformations proposed by Anagnostopoulos et al. (2006) are used. Two additional move operators, 'SwapByesMatches' and 'GroupingAwayUsingByes' (cf. supra), are proposed. These move operators are only applicable to the RTTP problem since they focus on the byes of a team.

[9] https://mat.tepper.cmu.edu/TOURN/

[10] http://www.sportscheduling.ugent.be/RobinX/travelRepo.php

The $RAIS_{TTP}$ algorithm is constructed as follows. First a single round-robin schedule is created using the polygon method, as described by Ribeiro and Urrutia (2007). To obtain the double round-robin the single round-robin is mirrored and venues are assigned randomly. After this initial schedule is constructed, the algorithm will follow a loop until the maximum of iterations is reached. An iteration starts with calculating the fitness of all available solutions. Next, the clonation rate is calculated to determine which percentage of the available solutions will be used and which percentage will be discarded. The remaining solutions will have the best fitness values and are therefore the closest to an optimal value. Next, a hypermutation is performed on all remaining solutions using the seven transformations. This will generate some new solutions that are potentially closer to an optimal objective function value. Next, a diversity selection is performed in order to select the most different solutions, in terms of home/away pattern, from the best ones. This step avoids that algorithm gets stuck in a local optimum. The last step of each iteration is generating some new solutions based on the existing ones. Together they will form the group of solutions for the next iteration. When the maximum number of iterations is reached, the best solution will be the solution that has the highest fitness function value. This method performs well compared to other methods but is not yet capable obtain a proven optimal schedule for instances consisting out of more than 10 teams.

3.5 Additional Readings

This section is used to cover some interesting additional readings that not necessarily belong to the field of (R)TTP. However, findings from these readings could be interesting to apply to the time-relaxed travelling tournament problem.

According to Schönberger et al. (2004) the travelling tournament problem (TTP) can be defined as 'temporally constrained'. Meaning that the number of matchdays is exactly enough to set up all fixtures. In the category of 'temporally relaxed', Schönberger et al. (2004) make a distinction between two different types. The first type of 'temporally relaxed' problems are those who have an extensive amount of rounds or timeslots and accordingly a considerable number of byes. An example of such problems are the scheduling problems in non-commercial sports leagues. Various methods have been established to solve these optimisation problems efficiently, e.g. combination of the tabu search and transportation problems (Van Bulck et al., 2019),the memetic algorithm (Schönberger et al., 2004; Van Bulck & Goossens, 2020) or a resource based approach (Knust, 2010).

The second type of 'temporally relaxed' problems include the problem studied in this master's dissertation. This second type is defined as the relaxed travelling tournament problem of competitions with 4 to 20 teams with up to 3 byes. This problem type can be seen as a limitation of the first type, where the number of playing rounds or timeslots are reduced to an almost absolute minimum.

A special remark should be made regarding the objectives of both types of problems. The first type with an excess of byes focuses primarily upon the rest days between fixtures of teams and a balanced assignment for all teams. Suksompong (2016) identifies three issues that should be handled when

scheduling such a tournament. First, periods with a high intensity (i.e. a lot of games in a short amount of time) or periods with low intensity can appear if the rest time between games has a high variation. Second, a difference in the rest time of 2 teams before their match-up can have an influence on the outcome of the game. A longer rest period could potentially create a positive bias for a team. Third, the difference in the number of games played per team can have an impact on the outlook of the classification after each time slot. 'Temporally relaxed' problems of the second type with few byes primarily focus on the total distance per team, especially in professional competitions. For most non-professional competitions, the distance has less impact on the schedule since most teams travel home after each game (Schönberger et al., 2004).

Van Bulck et al. (2019) proposed a new approach for a non-professional indoor football league scheduling, a time-relaxed competition with a large number of byes. Therefore, this problem could perhaps be compared to a RTTP problem with a large K and findings from this research can be used in RTTP problems where the value K is rather limited. Their approach is innovative since it implements venue and team availability into the scheduling.

Each team can present a set of timeslots where they are not able to play a game and a set of timeslots where they can play a home game. Based on this input data, the program strives for finding the best suited solution for each team. The method implements a heuristic approach, based on the principles of tabu search. For each non-tabu team a new transportation problem is solved to determine an optimal sequence of home games for this team and the optimal timeslots to play these games. After the optimisation is done for a team, this team is added to the tabu list. Once all teams are added to the tabu list, a new iteration can start. This new insight could be very interesting when the current RTTP is extended with availability constraints to closer match realistic scenarios. However, this approach is hard to implement on the current RTTP problem. Since this dissertation only works with a limited amount of byes ($K \leq 3$), not enough timeslots are available to change the match-ups of the different teams. Therefore, solving a transportation problem per team would probably not have a considerable impact on the solution of an RTTP instance.

Schönberger et al. (2004) created the Memetic Algorithm (MA) to solve 'temporally relaxed' problems of the first type. The Memetic Algorithm uses the fundamentals of Genetic Algorithms (GA) and adapts them to create solutions for non-commercial sports competitions. The results prove to be excellent for competitions with a huge number of available timeslots. When the number decreases, the search for an optimal solution becomes harder to almost impossible. In this master's dissertation, the focus is on RTTP with up to 3 byes, which can be seen as a special case of the problem studied by Schönberger et al. (2004). Their method, however, minimises other objectives while coping with availability constraints and is therefore less useful to solve the RTTP problem which limits the additional timeslots to an almost absolute minimum.

Another finding is the limited capability of genetic algorithms to solve both TTP and RTTP problems. This has been stipulated by Eiben and Ruttkay (1997) who claim that the blindness of genetic operators represents a major obstacle of applying Genetic Algorithms. The operators frequently introduce new constraint violations leading to an inferior mean solution quality. In turn,

this lowers the selection pressure over the course of the run. In the end, Genetic Algorithms have the tendency to converge prematurely without finding a solution of reasonable quality (Eiben & Ruttkay, 1997).

The next example of a 'temporally relaxed' problem is the non-professional table tennis league in Germany, investigated by Knust (2010). Here a special case of the multi-mode resource-constrained project scheduling problem (MRCPSP) with time-dependent resource profiles and partially renewable resources is created. In order to obtain a highly qualitative schedule, the problem is split into 2 subproblems. The first subproblem balances the home-away assignments of all teams. The second subproblem determines a feasible schedule for all matches in correspondence with the home-away assignments of the first subproblem. This procedure is repeated in multiple iterations with the principles of local search. To solve the second subproblem a genetic algorithm created for RCPSP problems by Hartmann (1998) is adapted to the specials needs of this subproblem in order for it to run more efficiently.

The construction of timetables can also be done using an adaptive large neighbourhood search (ALNS) as proposed by Van Bulck and Goossens (2020). Two steps should be covered each iteration. First a part of the schedule should be fixed, the other part can then be changed and optimised. The selection can be time- or team-based, where respectively a number of complete timeslots or a number of complete team schedules can be changed. The second stage consists of optimising the variable cells using IP . The new solution will be accepted in a similar way as new solutions are accepted in TTSA. This means that a new solution will always be accepted when the objective function value is improved or will be accepted with a certain probability when this is not the case. Next to ALNS, Van Bulck and Goossens (2020) suggest a second method for constructing schedules, named Memetic Algorithm. This approach can be compared to the one proposed by Schönberger et al. (2004) and will therefore not be further discussed here.

Another finding involves break minimisation and maximisation. The RTTP problem covered in this dissertation tries to maximise the number of breaks as mentioned in Section 2.2, hereby minimising the number of trips which in turn could lead to a lower travelling distance. Some, however, try to reach the opposite, break minimisation or a maximum of trips for each team. Régin (2001) implements constraint programming in order to minimise the number of breaks in a sports schedule. Trick (2000) also makes use of break minimisation in his schedule-then-break approach. This break minimisation can be useful for the RTTP problem. It has been shown by Miyashiro and Matsui (2005) that an optimal solution of the break minimisation problem can be constructed from that of the maximisation problem and vice versa. This means that a schedule with a minimum of $2n - 2$ breaks for $2n$ teams is essentially equivalent to a schedule with a maximum of breaks (Miyashiro & Matsui, 2005). This break maximisation schedule will return an optimal solution for RTTP instances with a constant distance (distance between all teams is the same) as shown by Suzuka, Miyashiro, Yoshise, and Matsui (2007).

Additionally, it is interesting to study the some of the mechanics behind the ALNS method in more detail, mainly the implementation of a dynamically adapting selection mechanism. Pisinger and Ropke (2010) use this kind of selection mechanism in order to determine which destroy/repair methods will be selected. However, it might be interesting to apply a similar method to

select between different move operators. This dynamic selection method will be described in detail in the next paragraph.

Each destroy/repair mechanism is assigned a weight that determines how often each particular method will be selected during the algorithm. The weights are adjusted dynamically to let the algorithm adapt to the instance at hand and the state of the search. Initially all methods are assigned the same weight ρ. The ratio of the weight of a method to the sum of all weights represents the probability that a method will be selected. Now the weights are adjusted dynamically based on the recorded performance of the corresponding method. Whenever the ALNS heuristic is completed, a score ψ is assigned based on the performance of the method. For example, the value of ψ is different if the method returns a new global best solution as opposed to when the method returns a solution that will only be accepted. The new weight of a destroy/repair method is then calculated using the following formula:

$$\rho = \lambda\rho + (1 - \lambda)\psi$$

where $\lambda \in [0, 1]$ represents the *decay* parameter. λ how sensitive the weights are to differences in the performance of the different methods.

Another interesting reading was the doctoral dissertation of Bao (2009) on time-relaxed round-robin tournaments and the NBA scheduling problem. This dissertation covers integer and constraint programming formulations in a time-relaxed environment. Even though formulations specifically for the K-RTTP are not provided, this dissertation of Bao (2009) served as an introduction in integer en constraint programming in time-relaxed timetabling. Finally, it is important to highlight the competitiveness of timetabling problems, a statement that is confirmed by the existence of international timetabling competitions[11]. While the competitive nature of this field makes it more appealing for researchers, it introduces problems related to comparison of different algorithms and heuristics. Schaerf and Di Gaspero (2006) state that in order to compare two algorithms it is necessary to specify the used instances as well as the compared features (e.g. quality of the objective function, speed,...). While this might ensure reproducibility, researchers can still take advantage of the so-called *Mongolian horde* approach, where they run as many trials as they can and report only the best of them (Schaerf & Di Gaspero, 2006).

3.6 Overview of Past Research

In order to provide a broad view on the current state of the research on (R)TTP, this final section of the literature overview delivers a short summary of the methods and results of past research. Also, some additional information is provided on the different instances that exist for the (R)TTP.

3.6.1 Summary of Applied Moves

Table 19 covers a short summary of all the used move operators by the previously presented solution algorithms.

A special remark should be made on the "Simulated annealing with hill climb" method created by Lim, Rodrigues, and Zhang (2006). This method only has one move operator called 'Local jump', exchanging 2 complementary matches in 2 rounds. This move operator replaces the first five operators mentioned in Table 19.

[11] https://www.itc2019.org

	CSM	TS	TIDA*	TTSA	PBSA	DFS*	ACO	VNS-CS	ANT HYP	L. HYP	CLONALG	LANGFORD
Swap Homes	✓	✓	-	✓	✓	-	-	Home-away Swap	-	✓	✓	✓
Swap Rounds	✓	✓	-	✓	✓	-	-	Round Swap	✓	✓	✓	✓
Swap Teams	✓	✓	-	✓	✓	-	-	Team Swap	✓	✓	✓	✓
Partial Swap Rounds	-	Swap Matches	-	✓	✓	-	-	Partial Round Swap	✓	Swap Matches	✓	✓
Partial Swap Teams	-	Swap Match Round	-	✓	✓	-	-	Games swap	✓	Swap Match Round	✓	✓
Swap Byes Matches	-	-	-	-	-	-	-	-	-	-	✓	-
Grouping Away Using Byes	-	-	-	-	-	-	-	-	-	-	✓	-
Shift move	-	-	-	-	-	-	-	-	✓	-	-	-

Table 19: Used Move Operators

CSM = Complete Search Method, which combines branch-and-bound, metaheuristics and dynamic programming (Brandão & Pedroso, 2014)
TS = composite-neighbourhood tabu search Di Gaspero and Schaerf (2007)
IDA*= iterative deepening A* for travelling tournament problem (Uthus, Riddle, & Guesgen, 2012)
TTSA = travelling tournament simulated annealing (Anagnostopoulos et al., 2006)
PBSA = population-based simulated annealing (Van Hentenryck & Vergados, 2007)
DFS* = combines iterative deepening A* with depth-first branch-and-bound (Uthus, Riddle, & Guesgen, 2009b)
ACO = ant colony optimisation (Uthus, Riddle, & Guesgen, 2009a)
VNS-CS = clustering search approach (Biajoli & Lorena, 2007)
ANT HYP = ant based hyper-heuristic (Chen et al., 2007)
L. HYP = new learning hyper-heuristic (Misir, Wauters, Verbeeck, & Vanden Berghe, 2009)
CLONALG = Clonal Selection Algorithm (Pérez-Cáceres & Riff, 2015)
LANGFORD = Improved Neighbourhood (Langford, 2010)

3.6.2 Problem Instances

Several different problem instance classes were defined for the (R)TTP. In this section all seven instance classes mentioned on Trick's Website[12] are covered. The main focus of this dissertation will lie on the NL instances, hence their detailed description in this section, since these are most frequently covered in previous research. However, some experimental results for other instances are also included in Section 5.

The NL-instances are proposed to represent the scheduling of the National League in Major League Baseball by Easton et al. (2001). Even though several changes have been made to the composition of the National League, this instance class remains as it was originally defined. The distance between two teams is the "air distance" between the city centers of the cities these teams are located in. The National League consists out of 16 teams, however smaller instances of the problem can be generated by limiting the instance to the first x teams. All NLx instances are therefore a part of the following set {NL4, NL6, NL8, NL10, NL12, NL14, NL16}. The list of participating teams can be found in Table 20, in that prescribed order. Consequently, NL4 will represent a competition between the Atlanta Braves, the New York Mets, the Philadelphia Phillies and the Montreal Expos.

1	ATL	Atlanta Braves	9	STL	St. Louis Cardinals
2	NYM	New York Mets	10	MIL	Milwaukee Brewers
3	PHI	Philadelphia Phillies	11	HOU	Houston Astros
4	MON	Montreal Expos	12	COL	Colorado Rockies
5	FLA	Florida Marlins	13	SF	San Francisco Giants
6	PIT	Pittsburgh Pirates	14	SD	San Diego Padres
7	CIN	Cincinnati Reds	15	LA	Los Angeles Dodgers
8	CHI	Chicago Cubs	16	ARI	Arizona Diamondbacks

Table 20: Teams NL-instance Class

Similar to the NL-instances, three other instance classes of this problem represent real life competitions. The NFL-instances are based on the air distance between the city centers from NFL teams. An important difference with the NL-instances is that the NFL-instances facilitate problems with up to 32 teams[13]. Another set of instances, the SUPER-instances, are derived from the rugby union league Super 14 (Uthus et al., 2009b). An important characteristic of this class is its geographical difference with the other classes. The SUPER-instances include teams that are located in Australia, New Zealand and South-Africa, so the competition consists out of three clusters. In the NL-instances the teams are more evenly geographically distributed. Finally, the BRA-instance class, proposed by Urrutia and Ribeiro (2004), covers the 24 teams in the main division of the 2003 edition of the Brazilian soccer championship.

Two other instances are defined from a more theoretical perspective. A first theoretical class is the CON-instance class, in this class the distance between any two venues is equal. For these instances, Urrutia and Ribeiro (2004) show that distance minimisation is equivalent with break maximisation. The second

[12] `https://mat.gsia.cmu.edu/TOURN/`

[13] `http://www.sportscheduling.ugent.be/RobinX/travelRepo.php`

theoretical class is the CIRC-instance class, where all venues are located on a circle. These instances are proposed since solving the TSP for a circular distance matrix is trivial. However, solving the TTP for the CIRC-instance class still remains challenging. A last instance class, the GAL-instance class, was proposed by Uthus et al. (2012). The galaxy-instances are different since they cover distances in a 3D-space that includes Earth and 39 other exoplanets. This 3D-space makes the problem set even more challenging. Additionally, this set allows for instances up to 40 teams.

3.6.3 Experimental Results

Most researched methods have developed a solution for some NLx instances, these instances can therefore be used to compare the performance of the different methods. A summary can be found in Table 22. A first look at Table 22 learns that most methods are able to find an optimal solution for competitions with up to 8 teams. Finding an optimal solution for competitions with 10 or more teams is clearly a lot harder for all methods. The explanation for this phenomenon is simply that more teams imply more options and thus more efficient methods will excel here. The best performing method is 'Population Based Simulated Annealing' by Van Hentenryck and Vergados (2007) since it has the current best solution for NL12, NL14 and NL16. This method is clearly capable of constructing a good solution for instances with a high complexity, which is an important characteristic of the RTTP. Based on these results, 'Population Based Simulated Annealing' seems to be the most promising method to use as a basis for this dissertation.

The second set of results are shown in Table 21 and have been created for the RTTP instances. Until now only Brandão and Pedroso (2014) and Pérez-Cáceres and Riff (2015) have presented new solutions for the RTTP instances. A summary of their results on the NLx instances can be found in Table 21. The table shows the best solutions found up until this moment for RTTP NLx instances with 0, 1, 2 or 3 byes for every team. All results for the NL4, NL6 and NL8 are given by Brandão and Pedroso (2014). For the bigger instances, solutions are calculated by Pérez-Cáceres and Riff (2015). Note that not all of these results are proven to be optimal.

	$K = 0$	$K = 1$	$K = 2$	$K = 3$	$K > 0$ Solution found by
NL4	8276	8160	8160	8044	(Brandão & Pedroso, 2014)
NL6	23916	23124	22557	22557	(Brandão & Pedroso, 2014)
NL8	39721	39128	38761	38670	(Brandão & Pedroso, 2014)
NL10	59436	59425	59373	59582	(Pérez-Cáceres & Riff, 2015)
NL12	110729	117680	119067	116082	(Pérez-Cáceres & Riff, 2015)
NL14	188728	209616	209317	205690	(Pérez-Cáceres & Riff, 2015)
NL16	261687	307125	300188	297426	(Pérez-Cáceres & Riff, 2015)

Table 21: Current RTTP Solutions

Instances	CSM	TS	TIDA*	TTSA	PBSA	SA + HC	VNS-CS	ANT HYP	L. HYP	CLONALG	LANGFORD	Best Solution
NL4	8276		8276	8276		8276	8276	8276	8276			8276
NL6	23916		23916	23916		23916	26588	23916	23916			23916
NL8	39721		39721	39721		39721	41928	40361	39721	39721		39721
NL10		59583	59436	59583		59821	65193	65168	59583	59910	59436	59436
NL12		111483		111248	110729	115089	120906	123752	112873	119174	112298	110729
NL14		190174		189766	188728	196363	208824	225169	196058	204082	190056	188728
NL16		270063		267194	261687	274673	287130	321037	279330	292203		261687

Table 22: Current NLx Solutions TTP

CSM = Complete Search Method, which combines branch-and-bound, metaheuristics and dynamic programming (Brandão & Pedroso, 2014)
TS = composite-neighbourhood tabu search (Di Gaspero & Schaerf, 2007)
IDA*= iterative deepening A* for travelling tournament problem (Uthus et al., 2012)
TTSA = travelling tournament simulated annealing (Anagnostopoulos et al., 2006)
PBSA = population-based simulated annealing (Van Hentenryck & Vergados, 2007)
DFS* = combines iterative deepening A* with depth-first branch-and-bound (DFB&B) (Uthus et al., 2009b)
ACO = ant colony optimisation (Uthus et al., 2009a)
VNS-CS = clustering search approach (Biajoli & Lorena, 2007)
SA+HC = Simulated annealing with hill climb (Lim et al., 2006)
ANT HYP = ant based hyper-heuristic (Chen et al., 2007)
L. HYP = new learning hyper-heuristic (Misir et al., 2009)
CLONALG = Clonal Selection Algorithm (Pérez-Cáceres & Riff, 2015)
LANGFORD = Improved Neighbourhood (Langford, 2010)

4 Methodology

In this thesis, the time-relaxed travelling tournament problem is approached from two different viewpoints. Section 4.1 approaches the problem from a heuristic point of view and covers the RPBSA (time-relaxed population based simulated annealing) heuristic. In Section 4.2, two exact optimisation methods are presented for the RTTP by means of constraint programming models.

4.1 Heuristic Approach for RTTP: RPBSA

The Population-Based Simulated Annealing method (PBSA) (Van Hentenryck & Vergados, 2007) leverages TTSA (Anagnostopoulos et al., 2006), a simulated annealing algorithm for the TTP. This dissertation focuses on adapting the TTSA to enable its usage in a time-relaxed environment, this adapted algorithm will be called RTTSA . Since experimental findings show that PBSA delivers added value, it is appended as an additional step together with the RTTSA. This is done to obtain comparable results with the current best known solutions for the RTTP-instances (Brandão & Pedroso, 2014; Pérez-Cáceres & Riff, 2015). In the rest of this dissertation, the term RPBSA is used to refer to the PBSA method with RTTSA as underlying simulated annealing algorithm.

While substantial effort is undertaken to adapt the TTSA algorithm to a time-relaxed setting, the components of PBSA are implemented as-is. Indeed, the framework of the PBSA algorithm makes no assumptions on the underlying simulated annealing algorithm, and thus needs no adaption when applying the RTTSA algorithm instead of the TTSA algorithm. It is sufficient to say that PBSA is used to enhance the performance of RTTSA. This section will first discuss the fitness evaluation of a given schedule (Section 4.1.1) and will then deliver an overview of the main concepts of RTTSA (Section 4.1.2). The next part of this section covers a short overview on the implementation of RPBSA (Section 4.1.3) and the generation of an initial schedule (Section 4.1.4). Finally, the different adaptations made to the TTSA algorithm are highlighted (Section 4.1.5). A more detailed overview of TTSA can be found in previous literature (Anagnostopoulos et al., 2006).

4.1.1 Fitness Evaluation

Since the ability to compare the fitness of two different schedules is an essential element of the model used in this master's dissertation, this subject requires some additional clarification first. In order to define the fitness value of any given schedule, a distinction between essential and non-essential constraints has been made. To improve clarity the five constraints mentioned in Section 2 are repeated here.

- C1: Each team plays exactly two times against all other teams
- C2: Each team plays exactly once at the home venue of all other teams
- C3: Each team has a total of K byes
- C4: The length of all home stands and away trips should range from length L to length U ("atmost constraint")
- C5: Two teams should not play each other twice in two consecutive rounds ("norepeat constraint")

The first three constraints (C1, C2, C3) remain essential constraints, i.e. no schedules that violate these constraints can be returned by the algorithm. Violating one or more of these constraints would no longer return a schedule of a DRR with K byes per team. Therefore, schedules that violate one or more of the first three constraints are beyond the scope of the RTTP. For the other two constraints (C4, C5), the story is different. While schedules that violate C4, C5, or both do not correspond to feasible solutions, these schedules satisfy the basic constraints of a DRR with K byes per team i.e. the algorithm can potentially return schedules that violate these two constraints.

In this dissertation, different degrees of feasibility are defined according to the number of violations (nbv) a schedule has. Feasible schedules comply with all constraints and have zero violations. If a schedule has at least one violation, this schedule will be called an infeasible schedule. However, part of the infeasible schedules will only have a limited number of violations, these are the semi-feasible schedules. These semi-feasible schedules are explicitly defined since for these schedules the feasibility can potentially be restored by means of a repair mechanism (cf. Section 4.1.5.3). Each time a single team violates constraint C4 or C5, one violation is counted. The schedule represented in Table 23 is an infeasible schedule with seven violations, since it violates C4 three times and C5 four times. Note, however, that the sequence '-4,0,4' for team 1 does not violate C5, since the games against the same opponent are separated by an intermediary bye. This implementation of non-essential or soft constraints may facilitate the search since the ability of exploring infeasible regions appears to be essential for the success of simulated annealing on the travelling tournament problem (Anagnostopoulos et al., 2006).

T/R	1	2	3	4	5	6	7	8	9	10	11
1	3	-2	-5	-6	-4	0	4	2	5	-3	6
2	6	1	0	-4	-5	-3	3	-1	4	6	5
3	-1	-5	-4	5	6	2	-2	0	-6	1	4
4	-5	6	3	2	1	-6	-1	0	-2	5	-3
5	4	3	1	-3	2	0	6	-6	-1	-4	-2
6	-2	-4	0	1	-3	4	-5	5	3	-2	-1

Table 23: NL4 3-RTTP Schedule

In this dissertation, the objective function formula proposed by Anagnostopoulos et al. (2006) is utilized.

$$C(S) = \begin{cases} totalDistance(S) & \text{if } S \text{ is feasible} \\ \sqrt{totalDistance(S)^2 + [w * f(nbv(S))]^2} & , \text{ otherwise} \end{cases}$$

For more details on this objective function, the reader is referred to Section 3.3.2.2 (or previous literature (Anagnostopoulos et al., 2006)).

4.1.2 Main Concepts RTTSA

Time-Relaxed Travelling Tournament Simulated Annealing (RTTSA) uses the same concepts and is structured in a similar way as its time-constrained counterpart (TTSA). This section will briefly cover these concepts to provide clarity on where the proposed adaptations will fit in.

RTTSA aims to perform a search in the neighbourhood of a given schedule in order to obtain a schedule with a better objective function value. The neighbourhood of a schedule S can be defined as the set of all possible schedules S' that can be obtained by performing one of the nine defined move operators (Section 3.4). The concepts of the first five move operators ($SwapHomes$, $SwapRounds$, $SwapTeams$, $PartialSwapRounds$, $PartialSwapTeams$) are already implemented in the TTSA algorithm for the time-constrained TTP, but required some adaptations in order to cope with byes (Section 3.4.1). Two other move operators ($GroupingAwayUsingByes$, $SwapByes$) are obtained from previous work on the RTTP (Pérez-Cáceres & Riff, 2015). Finally, building on top of the existing move operators, two new operators are proposed ($ExtendAwayTrip$, $FlipSchedule$).

In order to perform the neighbourhood search of a solution, a simulated annealing metaheuristic is used (Kirkpatrick et al., 1983). Pseudocode for the RTTSA algorithm is shown in Algorithm 1. Line 7 to line 43 make clear that the main structure of the proposed RTTSA algorithm is in line with a standard simulated annealing meta-heuristic. For a given temperature T, the algorithm selects one of the moves and generates the schedule S'. If S' has at least one violation and at most the number of allowed violations (in this case, S' is called semi-feasible) the repairSchedule method (Section 4.1.5.3) is applied in order to try restore the feasibility of schedule S'.

Next, the algorithm determines if the objective function value (cost) of the, potentially repaired, schedule S' is lower than the cost of schedule S or is lower than the cost of the best feasible or infeasible schedule encountered at that point in the algorithm. If this is the case, TTSA applies the move, if not, the move still has a certain probability of being accepted. Note that strictly infeasible schedules (nbv $>$ nbvAllowed) also can be accepted. This is done to ensure that infeasible regions are also covered during the search. However, when schedule S' is entirely equal to schedule S in terms of both distance and scheduled matches, the move will always be rejected (see Section 4.1.5.3 for more details on this scenario). Whenever a move gets accepted, intermediary variables nbf and nbi (new best feasible and new best infeasible) are used to indicate the value of the best feasible and best infeasible schedule at that time. The probability of accepting a non-improving move should decrease over time, this is implemented by adding the variable *counter*, which is incremented for every non-improving move and reset to zero when the best solution (feasible or infeasible) found so far has been improved. When *counter* reaches a predetermined upper limit the temperature T is updated to βT, (constant $\beta < 1$) and *counter* is reset to zero. With this step we enter a new phase of the algorithm. This phase is then repeated a predetermined number of times.

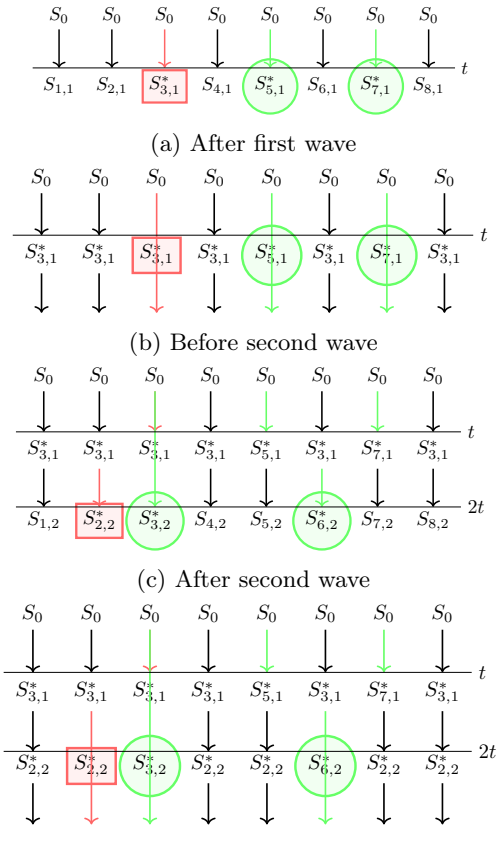
Some other concepts that were already applied in TTSA are the objective function mentioned earlier, the strategy for strategic oscillation and the usage of reheats. These concepts have not been adapted to be used in a time-relaxed environment. Consequently, the reader is referred to previous literature (Anagnostopoulos et al., 2006) for a detailed overview of these concepts.

Algorithm 1 RTTSA

```
1: procedure RTTSA(S_RPBSA)
2:     S ← S_RPBSA
3:     bestFeasible ← ∞, nbf ← ∞, bestInfeasible ← ∞, nbi ← ∞
4:     reheat ← 0, counter ← 0
5:     while reheat ≤ maxR do
6:         phase ← 0
7:         while phase ≤ maxP do
8:             counter ← 0
9:             while counter ≤ maxC do
10:                Move ← selectMove
11:                S′ ← Move(S)
12:                if 0 < nbv(S′) ≤ nbvAllowed then
13:                    S′ ← repairSchedule(S′)
14:                end if
15:                if C(S′) < C(S) or
                       nbv(S′) = 0 and C(S′) < bestFeasible or
                       nbv(S′) > 0 and C(S′) < bestInfeasible then
16:                    accept ← true
17:                else if ΔC = 0 and S′ = S then        ▷ ΔC = C(S′) − C(S)
18:                    accept ← false
19:                else
20:                    accept ← true with probability exp(−ΔC/T)
                                false otherwise
21:                end if
22:                if accept then
23:                    S ← S′
24:                    if nbv(S) = 0 then
25:                        nbf ← min(C(S), bestFeasible)
26:                    else
27:                        nbi ← min(C(S), bestInfeasible)
28:                    end if
29:                    if nbf < bestFeasible or nbi < bestInfeasible then
30:                        reheat ← 0, counter ← 0, phase ← 0
31:                        bestTemperature ← T
32:                        bestFeasible ← nbf
33:                        bestInfeasible ← nbi
34:                        if nbv(S) = 0 then
35:                            w ← w/θ
36:                        else
37:                            w ← w ∗ θ
38:                        end if
39:                    else
40:                        counter + +
41:                    end if
42:                end if
43:            end while
44:            phase + +
45:            T ← βT
46:        end while
47:        reheat + +
48:        T ← 2 ∗ bestTemperature
49:    end while
```

4.1.3 Implementation of RPBSA

In this dissertation, RTTSA is an intermediary step in the RPBSA algorithm. Since no adaptations have been made to the PBSA framework, a short overview of its implementation in this dissertation is sufficient. Figure 5 clarifies the mechanics of RPBSA. The initial solution (S_0), created using the 5.875-approximation method (Section 4.1.4), is used as a starting point. This initial schedule is copied a number of times (in the example 8 times) in order to generate a population. The next population is then obtained by applying RTTSA-algorithm to each member. This will result in a set of new schedules of which the best few are selected as the elite runs (in the example the best 3 are selected). These few elite runs will become member of the next population and the remaining slots of that population are filled with copies of the current best solution (indicated with a square in the example). Then the RTTSA-algorithm is applied once again. This is repeated during a predetermined number of phases. For a more detailed overview of the PBSA-algorithm, the reader is referred to previous literature (Van Hentenryck & Vergados, 2007).



(a) After first wave

(b) Before second wave

(c) After second wave

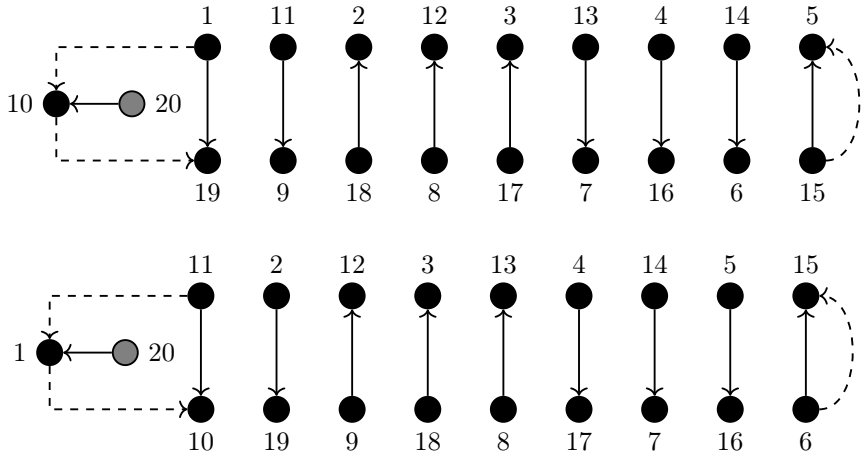
(d) Before third wave

(Based on Van Hentenryck and Vergados (2007))

Figure 5: Illustration RPBSA (Population Size 8 and 3 Elite) Runs

4.1.4 Initial Schedule

An initial schedule for the K-RTTP of n teams is constructed in two different steps. First the 5.875-approximation algorithm for the Travelling Tournament Problem, proposed by Westphal and Noparlik (2014), is used to generate a feasible schedule for a TTP instance of n teams (e.g. Table 24). While, Westphal and Noparlik propose their approximation algorithm for $U \geq 4$ and $n \geq 6$, their research clearly states how their method should be applied for other values of U and n. For $U = 3$, the orientation of certain arcs should be adapted (cf. Figure 6). In this case, the method guarantees an approximation ratio of $5/2 + 12/(n-1)$, which is not higher than 4.9 for any $n \geq 6$.



(cf. Westphal and Noparlik (2014))

Figure 6: Construction Round 1 and 2 ($n = 20, U = 3$)

The resulting time-constrained schedule is turned into a time-relaxed schedule by inserting K columns with byes (or zeros) in the last K rounds of the schedule, with K being the number of byes. This will result in an extended schedule where the last $2K$ rounds have the following pattern: round filled with matches, round filled with byes, round filled with matches, etc. Since some move operators are not able to shift byes, an additional step is added to make sure that all rounds at least have one match to be played. This is done by swapping x matches ($x \in [1, n/2]$) for every pair of a round full of matches and the following round full of byes. On average half of the matches of a round will be swapped, resulting in 2 consecutive rounds with half of teams playing in the first round and bye in the second and half of the teams having the opposite schedule. The sequence of matches remains unchanged and therefore the objective function value of the time-constrained schedule will be preserved. All this results in an initial schedule for the K-RTTP of n teams (e.g. Table 25, $n = 4, K = 2$).

T/R	1	2	3	4	5	6
1	2	3	-4	-3	4	-2
2	-1	4	-3	-4	3	1
3	4	-1	2	1	-2	-4
4	-3	-2	1	2	-1	3

Table 24: Initial NL4 TTP Schedule

T/R	1	2	3	4	5	6	7	8
1	2	3	-4	-3	4	0	0	-2
2	-1	4	-3	-4	0	3	0	1
3	4	-1	2	1	0	-2	-4	0
4	-3	-2	1	2	-1	0	3	0

Table 25: Initial NL4 RTTP with $K = 2$ Schedule

4.1.5 Adaptations to TTSA

This section highlights the adaptations that have been made to the TTSA algorithm. Some adaptations exploit the presence of byes and are only directly applicable for research on the RTTP. The changed move operators and the addition of a repair mechanism for semi-feasible instances, are examples of this. The decision to change the current uniform probability distribution of the move operators to a dynamic probability distribution, can be applied in both the time-relaxed and the time-constrained environment. This section will first discuss the adapted move operators and the new probability distribution. Finally the reparation of semi-feasible schedules is discussed.

4.1.5.1 Move Operators

4.1.5.1.1 Move operators TTSA and CLONALG

SwapHomes, *SwapRounds*, *SwapTeams*, *PartialSwapRounds* and finally the *PartialSwapTeams*-operator are the five move operators used in the TTSA algorithm. These require some adaptation in order to cope with a time-relaxed environment. The main objective is to avoid that the algorithm stops running when it encounters a bye. The necessary adaptations were already provided by Pérez-Cáceres and Riff (2015). These moves can change the candidate solution in many different ways, but none of them take the travelling distance explicitly into account. To guide the algorithm to better solutions in terms of total travelling distance, the two new moves introduced in the Clonal Selection Algorithm (Pérez-Cáceres & Riff, 2015) are also included, these moves are called *SwapByes* and *GroupingAwayUsingByes*. A detailed description of these seven move operators can be found in Section 3.4 and in previously mentioned literature.

4.1.5.1.2 New Move Operators

The two new move operators are *FlipSchedule* and *ExtendAwayTrip*. *FlipSchedule* is a drastic move operator that aims to radically change the candidate solution. This diversifying operator facilitates moving away from a local optimum in order to find a global optimum. *ExtendAwayTrip* slightly changes the *GroupingAwayUsingByes* operator by always adapting the schedule of the team that has to travel the most. This is implemented to increase the equitability of the schedules of all teams.

Flip Schedule

The objective function of a schedule $C(S)$ is mainly determined by the sequence of games all teams play. Inverting a complete schedule has no impact on the value of the objective function. However, inverting only part of the schedule potentially improves the objective function, since changes occur at the 'edges' of the inverted region. The move operator *FlipSchedule* is both applicable for TTP and RTTP instances. It can be seen as a series of *SwapRoundsRTTP* operators and as an adaptation of the $ShiftMove$ operator created by Chen et al. (2007). The method starts by randomly selecting 2 rounds, r_a and r_b ($r_a < r_b$). Hereafter, the positions of r_a and r_b and all intermediate rounds are inverted. This means that r_a will become r_b, $r_a + 1 \rightarrow r_b - 1$, $r_a + 2 \rightarrow r_b - 2$, etc. and finally r_b will become r_a. The pseudo code for this move operator is given in Algorithm 2. Table 26 provides an example of this move operator. The resulting schedule is not feasible since the no-repeat constraint is violated 4 times, in r_3 for t_1 and t_3, as well as in r_7 for t_1 and t_4.

Algorithm 2 Flip Schedule

```
1: procedure FLIP SCHEDULE(S)
2:     r_a < r_b | r ∈ R
3:     for m = 0, ..., ⌊(r_b − r_a)/2⌋ do
4:         S* ← SwapRounds(r_a + m, r_b − m)
5:     end for
6:     return S*
```

T/R	1	2	3	4	5	6	7	8	9
1	2	3	-4	0	0	0	-3	4	-2
2	-1	4	-3	-4	3	0	0	0	1
3	4	-1	2	0	-2	-4	1	0	0
4	-3	-2	1	2	0	3	0	-1	0

T/R	1	2	3	4	5	6	7	8	9
1	2	**3**	**-3**	0	0	0	**-4**	**4**	-2
2	-1	4	0	0	3	-4	-3	0	1
3	4	**-1**	**1**	-4	-2	0	2	0	0
4	-3	-2	0	3	0	2	**1**	**-1**	0

Table 26: Example $FlipSchedule(S, r_3, r_7)$

Extend Away Trip

The move operator *ExtendAwayTrip* combines an away trip of length 1 with one of length 2. First, the operator orders the teams in increasing order of the distance travelled. Second, it selects the first team having away trip of length 1, followed, after one or more home games (and possibly byes), by an away trip of length 2 (or vice versa). If such a team is found, 4 new schedules will be generated. If this team has multiple away trips of length 1 or 2, the rounds are selected randomly. The first schedule is generated by applying the move operator *PartialSwapRoundsRTTP*. The single away match will be swapped with the home match preceding the away trip of length 2. The second schedule is the result of a swap between the same matches but by using the move

operator $SwapRoundsRTTP$. The third generated schedule again makes use of $PartialSwapRoundsRTTP$ and swaps the single away match with the first home game following the 2 consecutive away games. The fourth and last schedule also inserts a single away game after the away trip op length 2 using $SwapRoundsRTTP$. In case the away trip of length 2 starts at round 0, only schedules 3 and 4 are generated. The opposite happens if the away trip of length 2 ends in the last round of the schedule, then the single away game is only insert in the round preceding the 2 consecutive away games. Finally, for all new schedules the cost is determined. As shown in Algorithm 3, the best schedule in terms of cost will be returned.

Algorithm 3 Extend Away Trip

```
1: procedure EXTEND AWAY TRIP(S)
2:     T* ← Team With Largest Distance
3:     r_a ← Round single away game
4:     r_b ← Round home game preceding away trip with length 2
5:     r_c ← Round home game succeeding away trip with length 2
6:     if r_a, r_b, r_c ∈ R then
7:         S_1 ← PartialSwapRoundsRTTP(T, r_a, r_b)
8:         S_2 ← SwapRoundsRTTP(r_a, r_b)
9:         S_3 ← PartialSwapRoundsRTTP(T, r_a, r_c)
10:        S_4 ← SwapRoundsRTTP(r_a, r_c)
11:    end if
12:    S* ← min(S_1, S_2, S_3, S_4)
13:    return S*
```

An example of how this move operator works is given in Table 27. First the team with the largest distance in the initial schedule is selected, in this case team 4. The selected single away game of team 4 takes place in round 1 and the selected away trip of length 2 takes place during rounds 4 and 5. Hereafter, four new schedules are generated by swapping matches in rounds 1 and 3 twice using $PartialsSwapRoundsRTTP$ and twice in rounds 1 and 6 using $SwapRoundsRTTP$. The cheapest schedule turns out to be the schedule generated using $SwapRoundsRTTP$ on rounds 1 and 6. This results in an away trip of length 3 for team 4 in rounds 3 to 5.

T/R	1	2	3	4	5	6	7	8	9	10	11
1	-5	0	5	4	-2	3	2	-4	-6	-3	6
2	4	0	-6	-5	1	6	-1	3	5	-4	-3
3	-6	0	-4	6	-5	-1	5	-2	4	1	2
4	-2	0	3	-1	-6	5	6	1	-3	2	-5
5	1	-6	-1	2	3	-4	-3	0	-2	6	4
6	3	5	2	-3	4	-2	-4	0	1	-5	-1

T/R	1	2	3	4	5	6	7	8	9	10	11
1	3	0	5	4	-2	-5	2	-4	-6	-3	6
2	6	0	-6	-5	1	4	-1	3	5	-4	-3
3	-1	0	-4	6	-5	-6	5	-2	4	1	2
4	5	0	3	-1	-6	-2	6	1	-3	2	-5
5	-4	-6	-1	2	3	1	-3	0	-2	6	4
6	-2	5	2	-3	4	3	-4	0	1	-5	-1

Table 27: Example $ExtendAwayTrip(S)$

4.1.5.2 Adaptation Select Move

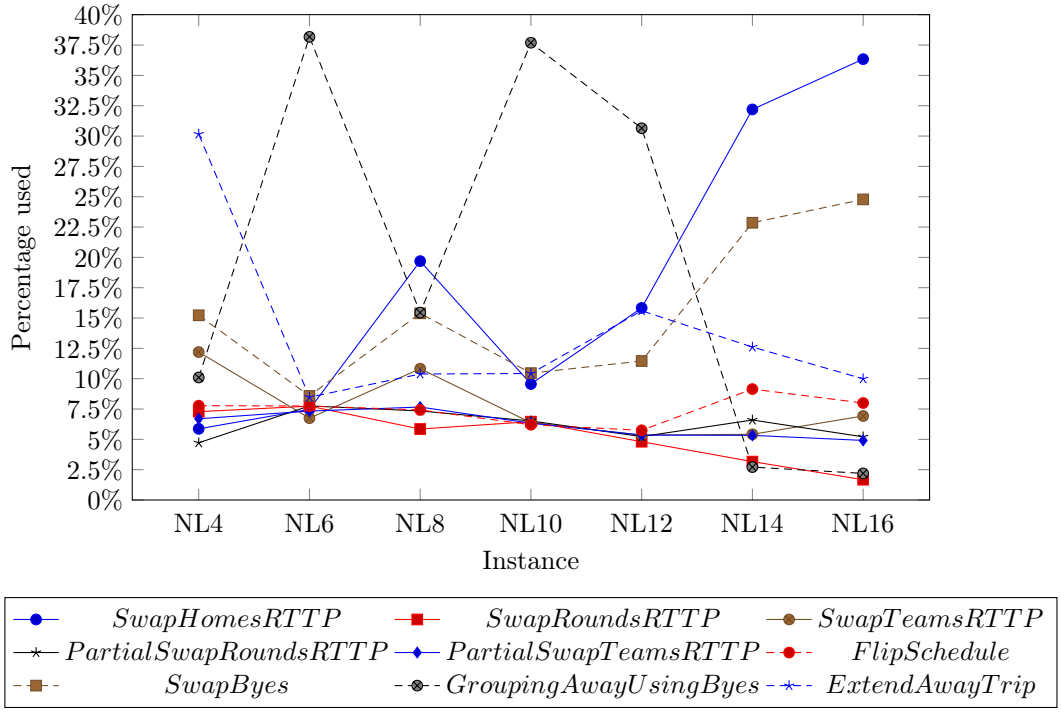
In the TTSA algorithm, the selection of the move operators follows a uniform probability distribution. The RTTSA algorithm steps away from this uniform distribution and uses a dynamic probability distribution which adjusts as the search progresses. First the need for an adapted probability distribution is studied. Running the RTTSA algorithm with a uniform distribution, while keeping track of how many improved schedules every move operator returns, resulted in clear differences in the 'success rate' of the different move operators.

The results of all runs are shown in Figure 7a and Figure 7b. Lines are selected to present the results in order to increase readability, however no options exist between the different x-values. For each NLx instance and each number of byes ($K \in \{1, 2, 3\}$), three runs of 30 minutes have been executed. Time-constrained instances are not considered in this experiment since not all move operators can be applied on these instances. Merging results for instances where $K = 0$ with the results for instances with other values of K could therefore lead to a distorted view of the actual probability distribution. Figure 7a shows the likelihood of each move operator conditioned on a successful move, i.e. a move that results in a lower value for the objective function $C(S)$. This likelihood is calculated as the ratio of the number of times a move operator has improved the solution to the total number of times the solution has been improved.

According to Figure 7a, which merges the results over different values for K for the same instance, $GroupingAwayUsingByes$ and $SwapHomesRTTP$ seem to be very promising moves, with overall percentages of 19.6% and 18.1%. For the larger instances with up to twelve teams, the operator $GroupingAwayUsingByes$ is dominant with probability peaks up to 40% for NL6 and NL10. $SwapHomesRTTP$, $SwapByes$ and $ExtendAwayTrip$ are the other good performers for these instances. Looking at the largest instances (NL14 and NL16) in Figure 7a, both $SwapHomesRTTP$ and $SwapByes$ perform particularly better than the other move operators, with a cumulative percentage for both operators of over 50%. Therefore, those 2 move operators are responsible for over half of the improvements found while running the algorithm. The dominant move operator $GroupingAwayUsingByes$ of the smaller instances seems to be less effective for the large instances, with less than 5%. The performance of the other 5 move operators is steady over all instances with an overall percentage between 5% and 10%. Since all move operators deliver some improving schedule, all operators can be considered useful in the search for a better solution.

Figure 7b, which merges the results over the different instances for different values of K, shows some similar results. $SwapHomesRTTP$, $SwapByes$, $GroupingAwayUsingByes$ and $ExtendAwayTrip$ are the dominant moves for all number of byes. An interesting trend is the decreasing performance of $SwapHomesRTTP$ as the number of byes increases. An opposite trend can be witnessed for $GroupingAwayUsingByes$. The performance of all other moves does not considerably vary over the different number of byes according to Figure 7b.

(a) Improving moves per instance



(b) Improving moves per byes

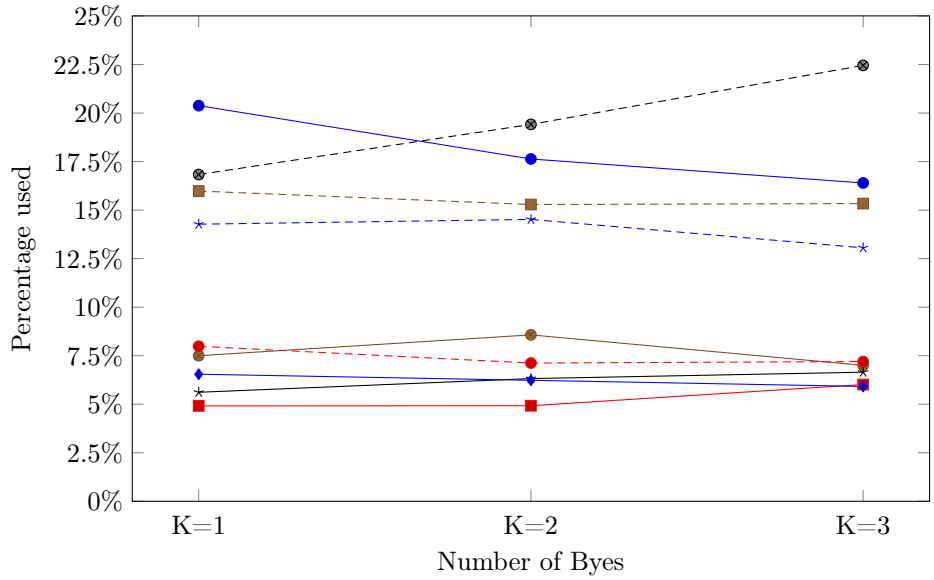


Figure 7: Move Operator Performance

Since the different move operators have a different 'success rate', i.e. some operators have a higher probability of returning a useful schedule than others, a uniform distribution of the move operators might no longer be ideal. A dynamic distribution, which adjusts as the search progresses, could improve the amount of useful schedules that the algorithm generates. The dynamic distribution proposed in this dissertation is based on the one proposed by Pisinger and Ropke (2010) in their study on the adaptive large neighbourhood search. A considerable difference is that instead of making adaptations to the weights of the different move operators, the adaptations are directly made to the probabilities. Each move operator j is assigned a probability of being selected ϕ_j, initially each operator has the same probability. Whenever a move operator is selected, the performance of this operator is given a score ψ.

Let a be the index of the move operator that was applied. A next step now consists of recalculating ϕ_a by adding ψ.

$$\phi_a = \phi_a + \psi$$

However, the sum of all probabilities ϕ_j will now exceed 1. This problem is solved by adapting all probabilities as follows

$$\phi_j = \frac{\phi_j}{1+\psi} \qquad \forall j$$

Now the sum of all probabilities will again be equal to 1. To prevent one move operator from being implemented too frequently, a final step which ensures that none of the probabilities will exceed 0.5 is added. The rationale behind this is that on average at least one in every two move operators should be different. Whenever a certain move operators has a probability that exceeds 0.5 after the calculation in the previous step, the excess probability is evenly redistributed over all other move operators. Let b be the index of a move operator with a probability that exceeds 0.5 with an excess of e and let N be the number of move operators.

$$\begin{cases} \phi_j = \phi_j + \frac{e}{N-1} & if \quad j \neq b \\ \phi_j = 0.5 & if \quad j = b \end{cases}$$

The value of ψ is computed using the following formula.

$$\psi = max \begin{cases} \omega_1 & \text{if the new schedule is a new best feasible} \\ \omega_2 & \text{if the new schedule is a new best infeasible} \\ \omega_3 & \text{otherwise} \end{cases}$$

The aim of this probability adaptation is to reward move operators that seem to work well for the instance at hand. Therefore, move operators returning new best feasible schedules should be rewarded more than operators that return a new best infeasible schedule. However, the latter case is still preferred over not returning a better schedule (feasible or infeasible) at all. Consequently, it is clear that in this case $\omega_1 > \omega_2 > \omega_3$ should hold. Since moves that do not improve the best (in)feasible schedule do not bring us closer to the goal, in terms of objective function value, ω_3 is assigned the value 0. To determine the values for ω_1 and ω_2, an assessment on the frequency of new better feasible or better infeasible schedules is made for different sizes of a competition. Figure 8 displays the mean absolute frequency of better schedules occurring over 9 runs of 30 minutes per instance size. Note, however, that a logarithmic y-axis is used to represent all results in one clear figure.

Figure 8 shows a sizeable fluctuation in the number of new best infeasible schedules generated over the different instances, ranging from 44 for NL4 to 4100 for NL10. This variation can not intuitively be covered by a function of the instance size and therefore a fixed value is assigned to ω_2. Several values were tested and the results of these tests are shown in Table 28. On average the results are best if ω_2 is set to 0.005% although this value is not the best for every instance. ω_1 is determined by again looking at the number of better feasible schedules generated. There seems to be a slight decreasing trend as instances get larger. This can be explained intuitively since the ratio of feasible schedules to the number of total produced schedules decreases with an increasing instance size. This means that for larger instances, the probability of producing a feasible schedule is smaller and accordingly the probability of producing a new better feasible schedule. To cope with this phenomenon, ω_1 should be an increasing function of the number of teams, i.e. feasible schedules in larger instances are rewarded more since their occurrence is less frequent. Figure 8 learns us that the differences between the instances are rather small, therefore chosen function is $\omega_1 = \sqrt{n} * x + \omega_2$. The addition of ω_2 ensures that $\omega_1 > \omega_2$ holds for any value of n and x.

To set the x value, the average feasible frequency is used. After 30 minutes, on average 300 better feasible schedules are generated or 10 per minute. In theory after 10 minutes 100 new feasible schedules would be found and for those 100, 100% should be distributed over the moves who generated those schedules. This results in 1% per new better feasible solution for the smallest instance NL4. Therefore x is set to 0.5% and the final value for ω_1 is set to $\sqrt{n}/200 + 0.00005$. While more extensive parameter tuning could potentially deliver a better result, the potential improvement of adapting these values will probably be minor. Therefore, the parameters ω_1, ω_2 and ω_3 are set at these intuitive values.

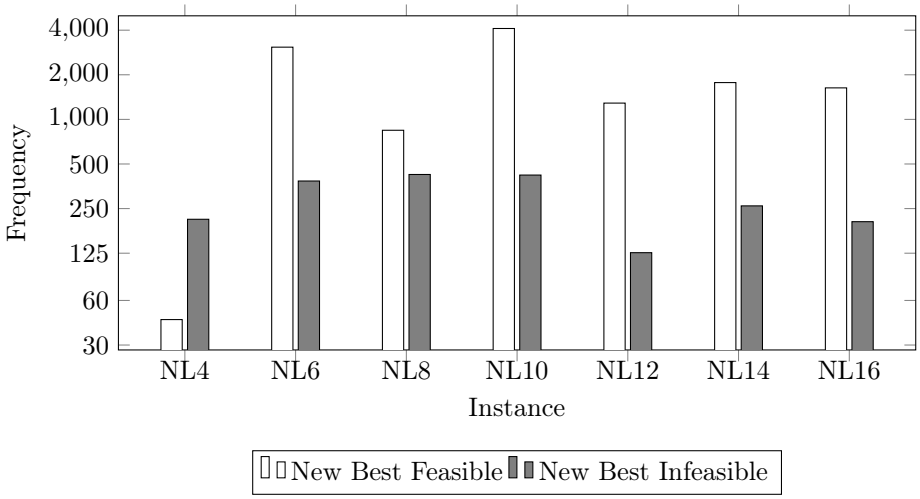


Figure 8: Frequency Better (In)Feasible Schedules

ω_2	0.10%	0.05%	0.01%	0.005%	0.001%
NL4K2	8160	8160	8160	8160	8160
NL6K2	23199	23301	22990	23374	23162
NL8K2	44183	43035	42898	42133	43127
NL10K2	71418	71851	67743	70339	70750
NL12K2	133902	133266	131353	130389	128231
NL14K2	260776	263268	260524	244638	258483
NL16K2	349283	341040	354091	350233	346990
Mean gap	12.93%	12.38%	11.35%	10.55%	11.41%

Table 28: Test Values ω_2

4.1.5.3 Repair Mechanism for Semi-feasible Schedules

The RTTSA algorithm is able to perform a large amount of moves, i.e. generate a lot of different schedules, with little computational effort. However, all move operators can potentially change a feasible schedule into an infeasible schedule, i.e. a schedule that violates one or more soft constraints (C4, C5). Table 29 gives an overview of which constraints can be violated by each move. A ⚠-icon shows that the constraint can potentially be violated when a certain move operator is applied to a feasible schedule.

Moves	C4	C5
SwapHomes	⚠	-
SwapRounds	⚠	⚠
SwapTeams	⚠	-
PartialSwapRounds	⚠	⚠
PartialSwapTeams	⚠	⚠
SwapByesMatches	⚠	⚠
GroupingAwayUsingByes	⚠	⚠
FlipSchedule	⚠	⚠
ExtendAwayTrip	⚠	⚠

⚠: Constraint can be violated when operator is applied

Table 29: Potential Violations (per move operator)

In fact, most of the generated schedules are not feasible, especially as the number of participating teams rises. The high percentage of infeasible solutions has been observed in time-relaxed sports timetabling problems with availability constraints in Schönberger et al. (2004) and Van Bulck and Goossens (2020). As a remedy, these authors propose to (partially) repair candidate solutions after applying move operators. Even though Schönberger et al. and Van Bulck and Goossens study a slightly different problem, the approach of repairing candidate solutions can also be useful for the RTTP.

Schedules with an excessive amount of violations require more computational effort to repair than schedules with a small number of violations. Schedules with more participating teams have on average more violations. Consequently the number of allowed violations should ideally be a multiple, denoted by τ, of the number of teams n, yielding $\lceil \tau n \rceil$, where the ceiling function ensures an integer

result. To determine this quantity for semi-feasible schedules, an experimental study is performed. Before discussing the results however, firstly the algorithm of the repair mechanism is covered. The pseudo code for this mechanism can be found in Algorithm 4.

Each time a move operator creates a new schedule, this schedule is checked for violations. If the new schedule is semi-feasible, the repair method is applied to try to restore the feasibility. First, an assessment is made to determine the number and location of violations. Basically, two types of violations are possible:

- Home stands or away trips exceeding 3 matches (C4 is violated)
- Two consecutive games against the same opponent (C5 is violated)

The assessment will create an array with the following information:

- Number of violations against C4
- Number of violations against C5
- Location of each violation (team and round of violation)

Once the assessment of violations has been executed, the repair method is called with the information on all violations as input. First, violations against the atmost constraint are trying to be repaired. The algorithm searches for a bye in the schedule and applies a $PartialSwapRoundsRTTP$ move operator. If the selected bye is not part of the trip, this move will swap it with each of the rounds that are part of the away trip or home stand that is exceeding the upper bound of 3. To clarify, consider the following pattern with an away trip of length 4: +,-,-,-,-,+ , if one of those away games is replaced by a bye, feasibility is restored. The resulting pattern can be as follows: +,-,0,-,-,+.

If a bye is part of the trip, a solution for this violation can often be found by partially swapping all rounds not part of the trip with this bye. An example of this can be seen in the following pattern: +,-,-,0,-,-,+ which is an away trip of length 4, by swapping the bye with a home game the pattern is changed to: +,-,-,+,-,-,+ which is feasible and has two away trips of length 2.

If the violation is repaired in the resulting schedule, this repaired schedule will be the new schedule. In case multiple feasible schedules can be created, the schedule that performs best in terms of total travelling distance is selected. If a violation against the upper bound of a trip is repaired, a new assessment is made of the violations in the schedule. If the violation is not repaired, the algorithm continues and tries to repair the remaining violations.

Violations against the no-repeat constraint per definition come in pairs. Two teams will have a no-repeat violation in the same round for consecutively playing the same opponent. To repair this violation, both matches against the same opponent of one team are swapped using the move operator $PartialSwapRound$ with all other rounds of this team. The best feasible schedule in terms of total travelling distance will be returned by the algorithm. If the no-repeat violation is repaired, then the no-repeat violation for the opponent in the same round will be repaired as well. Hereafter, a new assessment of the violations is made and the program can continue to resolve violations. If the violation is not repaired, the algorithm continues and tries to repair the remaining violations. To select which schedule the repair mechanism should return, the cost of both the input schedule and the resulting schedule are determined. The best schedule will be returned by the method.

Algorithm 4 Repair Schedule

```
1: procedure REPAIR SCHEDULE(S)
2:     violationInfo[] ← violationAssesment(S)
3:     U = set of violations against C4
4:     G = set of violations against C5
5:     for all u ∈ U do
6:         t ← Team of u
7:         b ← Round where t has a Bye
8:         if b ∈ R_u then                ▷ R_u = all rounds ∈ away trip/home stand
9:             for all r ∈ R \ R_u do
10:                S_r ← PartialSwapRounds(t, b, r)
11:        else
12:            for all r ∈ R_u do
13:                S_r ← PartialSwapRounds(t, b, r)
14:        if u is repaired then
15:            S* ← min(S_r)
16:            violationInfo[] ← violationAssesment(S*)
17:            Update U and G
18:    for all g ∈ G do
19:        t ← Team responsible for violation g
20:        z ← Round where t has a repeat game violation
21:        for all r ∈ R \ {z, z − 1} do
22:            S_r1 ← PartialSwapRounds(t, z, r)
23:            S_r2 ← PartialSwapRounds(t, z − 1, r)
24:        if g is repaired then
25:            S* ← min(S_r1, S_r2)
26:            violationInfo[] ← violationAssesment(S*)
27:            Update U and G
28:    return min(S*, S)
```

In order to determine the set of semi-feasible schedules, i.e. determine the maximum number of allowable violations, an experimental study was performed. Seven different values for the multiple were considered $\{0, 0.25, 0.50, 0.75, 1, 1.25, 1.50\}$ and were tested on four different instances {NL8,NL10,NL12,NL14} with three different number of byes, $K = \{1, 2, 3\}$. When for a certain instance, τn does not return an integer number of allowable violations, this number is rounded to the closest integer. To avoid excessive runtimes in this experiment, each run was terminated when a population was generated that did not improve the objective function value, compared to the previous population. All of the 84 combinations where considered three times in order to obtain representative mean values. The results of this experiment are summarised in Tables 30 and 31, and Figure 9. Since the difference between the different number of byes for the same instance are negligible for the purpose of this experiment, all results with the same instance and the same τ are merged.

In Table 30, the obtained results are represented relatively to the current best found solution for the different instances (Table 21). This is done to allow for comparability across the different instances. For example, using $\tau = 0$ on the instance NL8, returns a mean objective function value which lies 21.04% higher than the current best found solution for the same instance, with a mean runtime of 2.03 minutes. It is interesting to see that the gap is clearly smaller for instances where n is multiple of 4. Closing the gap to an optimal solution seems to be easier for these instances, something that was also noticed by Thielen and Westphal (2010) who derived a lower approximation ratio for instances with those n values.

The results in Table 30 are then used to construct Figure 9. This figure plots the average gap against the average runtime over the different instances for several values of τ. Repair multiple values $\{0.25, 1.25, 1.50\}$ are all dominated by others since they have nearly the same average gap for a higher runtime, or they have a higher average gap for the same runtime. The case $\tau = 0$ is not clearly dominated by one of the other multiples, but this is mainly due to the design of this experiment. A run terminates when a population does not improve the objective function value. With $\tau = 0$, no schedules are being repaired, therefore finding a better objective function value is becoming harder and a run will terminate sooner. The design of this experiment favours the average runtime when $\tau = 0$. The findings from Section 5 (cf. infra) also support the decision to only consider τ values larger than 0. This means that only three options for τ remain, $\{0.50, 0.75, 1\}$. Since these three values deliver comparable results on both the average runtime and the average gap, the conservative decision is made to average out these values. This results in fixing repair multiple τ to 0.75 when deciding the number of allowable violations. Therefore, a schedule of a competition with 10 teams will be semi-feasible when this schedule has up to 8 ($\lceil \tau n \rceil = \lceil 0.75 * 10 \rceil = \lceil 7.5 \rceil = 8$) violations.

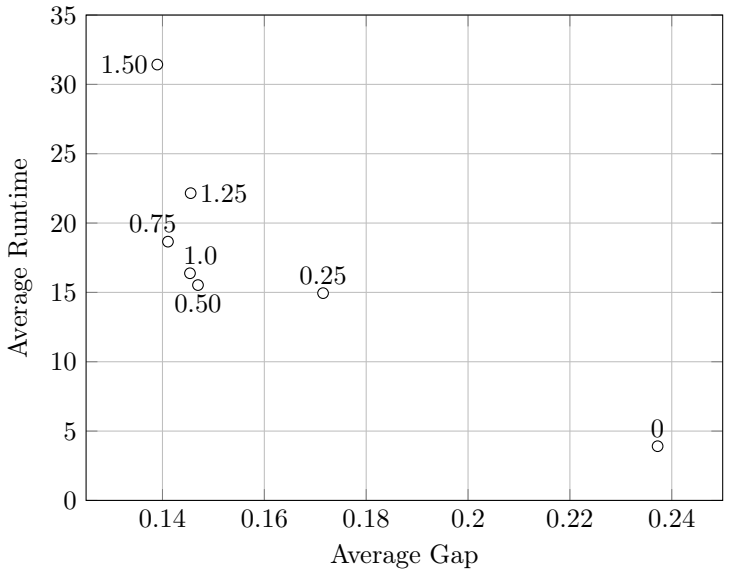


Figure 9: Gap - Runtime Plot

	τ	0	0.25	0.50	0.75	1.0	1.25	1.50
NL8	Mean gap to optimal	21.04%	14.02%	10.97%	9.80%	9.73%	9.21%	7.81%
	Mean time (in minutes)	2.03	3.8	5.2	8.94	14.61	20.12	35.39
NL10	Mean gap to optimal	23.65%	19.17%	16.63%	17.25%	17.03%	17.63%	16.80%
	Mean time (in minutes)	3.44	8.61	10.67	8	13.1	19.3	34.74
NL12	Mean gap to optimal	16.93%	11.45%	10.15%	10.27%	11.22%	11.18%	11.50%
	Mean time (in minutes)	4.05	11.3	16.96	19.65	12.64	16.19	19.68
NL14	Mean gap to optimal	33.27%	23.97%	21.03%	19.10%	20.18%	20.20%	19.48%
	Mean time (in minutes)	6.12	36.07	29.28	38.05	25.18	33.01	35.91

Table 30: Mean Gap and Runtime (per instance per τ)

Even though certain adaptations have been made to the algorithm in order to perform this experiment, the results still deliver a decent indication on how capable the mechanism is to return better (and feasible) schedules. Again the results can be merged for all instances with the same number of teams and the same τ. Table 31 summarizes this capability using two variables, repaired used and repaired feasible. Repaired used covers the instances where the repair chain returns a schedule with a lower objective function. If this better schedule is feasible, then it also counted as repaired feasible.

	τ	0.25	0.50	0.75	1.0	1.25	1.50
NL8	Repaired used	77.33%	77.22%	73.11%	79.33%	84.78%	82.67%
	Repaired feasible	70.37%	63.28%	56.62%	57.38%	58.78%	56.21%
NL10	Repaired used	64.02%	74.40%	82.26%	86.67%	88.24%	89.50%
	Repaired feasible	52.70%	52.57%	54.14%	52.73%	51.36%	49.77%
NL12	Repaired used	48.00%	48.33%	60.00%	72.00%	82.33%	87.67%
	Repaired feasible	40.19%	33.83%	36.67%	37.20%	37.42%	37.60%
NL14	Repaired used	43.00%	39.33%	43.33%	44.44%	52.89%	65.33%
	Repaired feasible	33.21%	26.75%	24.12%	20.30%	21.10%	23.52%

Table 31: Performance Repair Mechanism

Since $\tau = 0.75$ is selected, only these results are studied in detail. The number of feasible schedules returned by the repair mechanism is clearly dependent on the size of the problem instance. This seems quite logical, since instances with more teams have bigger schedules and the average number of violations is higher for bigger schedules. The repair mechanism will return a better schedule, in terms of objective function value, in at least 40% of the cases. This clearly shows that implementing this repair mechanism might improve the overall performance of the algorithm. Another observation is the decrease, on average, of *Repaired feasible* with increasing values of τ. Since increasing τ allows more violations in semi-feasible schedules, the intuitive reasoning that schedules with more violations are harder to repair is confirmed.

However, in exceptional circumstances the repair mechanism could undo the move that was performed by one of the operators. Schedules are in that case repaired to the schedule as it was before applying the move operator. To avoid cycles and slowing down the algorithm significantly, an additional check is implemented. This check is implemented on line 17 of Algorithm 1 on Page

42. Whenever the new, potentially repaired, schedule has the same objective function value as the previous schedule, these two schedules are compared. If both schedules are in fact entirely equal, the move will not be accepted. This computationally expensive complete comparison of two schedules is therefore only executed when problems can arise.

4.2 Constraint Programming for RTTP

While heuristic algorithms, like the RPBSA, potentially deliver high quality solutions within a reasonable timeframe, these solutions are not necessarily an optimal solution. Discrete optimisation strategies will always return an optimal solution, given enough time. Therefore, this thesis also introduces two discrete optimisation formulations for the K-RTTP specifications at hand. Note that other, potentially better performing, models can be drawn up.

Previous research on discrete optimisation for time-relaxed problems is quite limited. To the best of our knowledge, only Bao (2009) proposed several IP- and CP-formulations for the RTTP. However, in his work Bao mainly focuses on break minimisation and time-relaxed problems with considerably more timeslots. Consequently, the exact problem studied in this dissertation is only briefly covered by Model 22 in Bao (2009). However, since this model was only proposed as a comparison with models for other problem specifications, a new CP-formulation specifically proposed for the problem at hand could potentially improve the performance. The work of Bao (2009) could still be used to provide valuable insights in IP- and CP-modelling for time-relaxed problems. As mentioned in Section 3.3.3, both the IP- and the CP-formulation have their advantages. However, since this is, to the best of our knowledge, the first proposal of IP- or CP-models for solving the specific K-RTTP specifications of this thesis, an intuitive approach is followed in drawing up these models. The modelling framework offered by constraint programming is more flexible than the framework provided by integer programming and is therefore seen as more suited for drawing up an intuitive representation of the problem. Additionally, Model 22 by Bao (2009) was also proposed as a CP-model. A more in depth discussion on the modelling decisions and there potential alternatives is provided in Section 4.2.3.

This section will first provide a CP-formulation specifically for the K-RTTP specifications in this dissertation. In the second part of this section, a CP-formulation is proposed for the time-relaxed timetable constrained distance minimisation problem (RTCDMP). For any given opponent schedule, the solution of this problem returns an optimal home-away assignment, given enough time. Consequently, the RTCDMP can be used as the second step in constructing a time-relaxed double round-robin tournament schedule when following a first-schedule-then-break approach. However, the RTCDMP can also be used to further enhance the results of any RTTP-algorithm. Using the opponent schedule, provided by any given RTTP-algorithm, for the RTCDMP, should result in an optimal home-away assignment for this opponent schedule. Therefore, the RTCDMP can either prove that the solution delivered by the RTTP-algorithm can not be further improved by adapting the home-away assignment or it can return an improved schedule with an optimal home-away assignment. Finally, the last part of this section covers a brief discussion on the modelling decisions. Note, that n still represents the number of teams.

4.2.1 RTTP

In the following CP-model for the RTTP, T denotes the set of teams, while R denotes the set of rounds and $R^0 = R \cup \{0\}$. The distance matrix is represented by D, and $D_{i,j}$ contains the distance between the venues i and j. Finally, the number of byes is represented by B and Ub denotes the upper bound on the number of consecutive home or consecutive away games. To formulate the problem, a $n \times n$ matrix x is introduced, where the elements x_{ij} equal the round r in which team i plays a home game against team j in round r, and x_{ii} equals zero. Integer variable v_{ir} for each $i \in T$ and each $r \in R^0 \cup \{|R| + 1\}$ is used to indicate the venue where team i is located in round r. Dummy slots 0 and $|R| + 1$ are used to make sure every team starts and finishes the competition at their own venue. The travel distance of team i between round r and $r + 1$ is represented by $D_{v_{ir},v_{i(r+1)}}$. This results into the CP-model on the next page.

The objective function is given in (1), the goal of the RTTP is to minimise the travelled distance by all teams over all rounds. The alldifferent constraint (2) states that every team should at most play one home or away match in every round. Constraints (3) indicate that a team should not play against itself. Venues are assigned for all rounds by constraints (4). Constraints (5) and (6) state that every team starts and ends the competition at their own venue. Constraints (7) ensure that when a team has a bye, i.e. it has no matches in a round, the team will stay at the venue of the previous round. The no-repeat constraint is enforced by adding constrains (8). Constraints (9) and (10) set an upper bound to the number of consecutive home and consecutive away games.

Since these two constraints are not straightforward, they might require some additional clarification. First, let us consider constraints (9), these constraints limit the number of consecutive home games. When $Ub = 3$, at most three consecutive home games are allowed. In a time-constrained environment, these allowed home stands will also appear in three consecutive rounds. However, in a time-relaxed environment byes can be present in a home stand, which means that an allowed home stand can now be spread over more than three rounds. When setting $B = 2$, allowed home stands can also take place in four or five consecutive rounds. This explains why the constraints are added for every value of $b\,(b \in \{0, \ldots, B\})$. Constraints (9) are proposed in an IF-THEN form. If team i plays no away games during $Ub + b$ rounds, then team i can have at most Ub home games in that same period. This condition is necessary since byes are ignored when determining the length of home stands and away trips. Constraints (10) operate exactly the same, but they limit the number of consecutive away games. Together with constraints (17) and (18), which add the proper limitations to the different decision variables, these first ten constraint types are sufficient for modelling the problem.

However, by adding some, at first sight unnecessary, constraints, the performance of a CP-model can potentially be improved. These so-called redundant constraints do not further restrict the solution space, but they can potentially reduce the search space. Consequently, the efficiency of the CP-model is improved since less potential schedules have to be considered. These redundant constraints express properties of the problem solutions that are not explicitly listed in the first ten constraints.

$$\text{min:} \quad \sum_{i \in T} \sum_{r \in R^0} D_{v_{ir}, v_{i(r+1)}} \tag{1}$$

$$\text{s.t.} \quad alldifferent(x_{ij}, x_{ji} : j \in T, j \neq i) \qquad \forall i \in T \tag{2}$$

$$x_{ii} = 0 \qquad \forall i \in T \tag{3}$$

$$x_{ij} = r \Rightarrow v_{ir} = i \wedge v_{jr} = i \qquad \forall i, j \in T, \forall r \in R \tag{4}$$

$$v_{i0} = i \qquad \forall i \in T \tag{5}$$

$$v_{i(|R|+1)} = i \qquad \forall i \in T \tag{6}$$

$$\sum_{j \in T} (x_{ij} \vee x_{ji} = r) = 0 \Rightarrow v_{ir} = v_{i(r-1)} \qquad \forall i \in T, \forall r \in R \tag{7}$$

$$|x_{ij} - x_{ji}| > 1 \qquad \forall i, j \in T : i < j \tag{8}$$

$$\sum_{p=0}^{Ub+b} \sum_{j \in T} (x_{ji} = r + p) = 0 \Rightarrow \sum_{p=0}^{Ub+b} \sum_{j \in T} (x_{ij} = r + p) \leq Ub \qquad \begin{array}{l} \forall i \in T, \forall b \in \{0, \ldots, B\}, \\ \forall r \in \{1, \ldots, |R| - Ub + b\} \end{array} \tag{9}$$

$$\sum_{p=0}^{Ub+b} \sum_{j \in T} (x_{ij} = r + p) = 0 \Rightarrow \sum_{p=0}^{Ub+b} \sum_{j \in T} (x_{ji} = r + p) \leq Ub \qquad \begin{array}{l} \forall i \in T, \forall b \in \{0, \ldots, B\}, \\ \forall r \in \{1, \ldots, |R| - Ub + b\} \end{array} \tag{10}$$

$$\sum_{p=0}^{Ub+B} \sum_{j \in T} (x_{ij} = r + p) \geq 1 \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - B\} \end{array} \tag{11}$$

$$\sum_{p=0}^{Ub+B} \sum_{j \in T} (x_{ji} = r + p) \geq 1 \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - B\} \end{array} \tag{12}$$

$$\sum_{r=1}^{l} \sum_{j \in T} (x_{ij} = r) \geq n - 1 - k * Ub \qquad \begin{array}{l} \forall i \in T, \\ \forall k \in \{1, \ldots, n-1\} \\ l = |R| - k * (Ub + 1) \end{array} \tag{13}$$

$$\sum_{r=1}^{l} \sum_{j \in T} (x_{ji} = r) \geq n - 1 - k * Ub \qquad \begin{array}{l} \forall i \in T, \\ \forall k \in \{1, \ldots, n-1\} \\ l = |R| - k * (Ub + 1) \end{array} \tag{14}$$

$$\sum_{i \in T} \sum_{j \in T \setminus \{1\}} (x_{ij} = r) \leq n/2 \qquad \forall r \in R \tag{15}$$

$$x_{1,2} > x_{2,1} \tag{16}$$

$$x_{ij} \in R^0 \qquad \forall i, j \in T \tag{17}$$

$$v_{ir} \in \{1, \ldots, n\} \qquad \forall i \in T, \forall r \in R^0 \cup \{|R| + 1\} \tag{18}$$

Constraints (11) (and (12)) ensure that teams play at least one home (away) game in Ub+1+B consecutive rounds. These constraints are further extended by constraints (13) (and (14)). If a team can play at most Ub home (away) games in Ub+1 consecutive rounds, a team can only play Ub home (away) games in the last Ub+1 rounds. Therefore, at least $n - 1 - Ub$ home (away) games should be scheduled in all rounds preceding the last Ub+1 rounds. This finding can then be extended to the last $2Ub + 1, 3Ub + 1, \ldots$ rounds. And finally constraints (15) state that every round can have at most $n/2$ games. Another important characteristic of solutions for this problem is their potential symmetry. For any given schedule, the matches can be played in the opposite order and the resulting schedule will return the same objective function value. Therefore, only one of these two schedules should be considered, constraint (16) states that all schedules should place the home match of team 1 against team 2, before the away match of team 1 against team 2. By taking advantage of this symmetry, the size of the solution space it reduced to half of its original size. This can be seen as a technique to break the reflective symmetry as proposed by Uthus et al. (2012).

The discussed formulation is further clarified by its OPL code in Appendix A. In this code the implementation of the variable selection strategy is also made clear. Since the v_{ir} variables are mainly determined by the value of the x_{ij} variables, it is clear that the latter variables should be fixed first.

4.2.2 RTCDMP

The time-relaxed timetable constrained distance minimisation problem is defined similar to how Rasmussen and Trick (2008) defined the TCDMP for time-constrained problems. The RTCDMP still consists out of finding an optimal home-away assignment, in terms of distance minimisation, for any given opponent schedule. However, this opponent schedule will now include one or more byes per team. While this seems like a minor adjustment, the impact on the complexity of the problem formulation is considerable. In fact, this problem is an adaptation of the problem studied in Section 4.2.1, where the opponent schedule is already fixed. Again, a more intuitive modelling approach is preferred for the scope of this master's dissertation.

In the following CP-model, T again denotes the set teams, while R denotes the set of rounds and $R^0 = R \cup \{0\}$. The number of byes is indicated by B and Ub sets the upper bound for the number of consecutive home or consecutive away games. The distance matrix is represented by D, and D_{ij} equals the distance between the venues of team i and team j. The opponent schedule or time table is represented by TT, the opponent of team i in round r is indicated by TT_{ir} ($TT_{i,r} = 0$ if team i has a bye in round r). In order to formulate this CP-model, two integer decision variable types are defined. To indicate whether a team plays at home, plays away or has a bye, decision variable h_{ir} is introduced for each $i \in T$ and for each $r \in R$. h_{ir} equals 0 if team i plays away in round r, 1 if team i plays at home in round r and, similar to Bao (2009), 2 if team i has a bye in round r. Integer variable v_{ir} for each $i \in T$ and each $r \in R^0 \cup \{|R| + 1\}$ indicates the venue where team i is located in round r. Dummy slots 0 and $|R| + 1$ are used to make sure every team starts and finishes the competition at their own venue. $D_{v_{ir}v_{i(r+1)}}$ indicates the distance team i travels between round r and round $r + 1$.

This all results in the following CP-formulation for the RTCDMP.

$$\text{min:} \quad \sum_{i \in T} \sum_{r \in R^0} D_{v_{ir} v_{i(r+1)}} \tag{19}$$

s.t.

$$TT_{ir} = 0 \Rightarrow h_{ir} = 2 \wedge v_{ir} = v_{i(r-1)} \qquad \forall i \in T, \forall r \in R \tag{20}$$

$$\sum_{r \in R} (h_{ir} = 1) = n - 1 \qquad \forall i \in T \tag{21}$$

$$\sum_{r \in R} (h_{ir} = 0) = n - 1 \qquad \forall i \in T \tag{22}$$

$$TT_{ir} \neq 0 \Rightarrow h_{ir} + h_{TT_{ir}r} = 1 \qquad \forall i \in T, \forall r \in R \tag{23}$$

$$TT_{ir_1} = TT_{ir_2} \wedge TT_{ir_1} \neq 0 \Rightarrow h_{ir_1} + h_{ir_2} = 1 \qquad \forall i \in T, \forall r_1, r_2 \in R \tag{24}$$

$$h_{ir} = 0 \Rightarrow v_{ir} = TT_{ir} \qquad \forall i \in T, \forall r \in R \tag{25}$$

$$h_{ir} = 1 \Rightarrow v_{ir} = i \qquad \forall i \in T, \forall r \in R \tag{26}$$

$$v_{i0} = i \qquad \forall i \in T \tag{27}$$

$$v_{i(|R|+1)} = i \qquad \forall i \in T \tag{28}$$

$$\sum_{a=0}^{Ub+b} (h_{i(r+a)} = 0) = 0 \Rightarrow \sum_{a=0}^{Ub+b} (h_{i(r+a)} = 1) \leq Ub \qquad \begin{array}{l} \forall b \in \{0, \ldots, B\}, \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - b\} \end{array} \tag{29}$$

$$\sum_{a=0}^{Ub+b} (h_{i(r+a)} = 1) = 0 \Rightarrow \sum_{a=0}^{Ub+b} (h_{i(r+a)} = 0) \leq Ub \qquad \begin{array}{l} \forall b \in \{0, \ldots, B\}, \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - b\} \end{array} \tag{30}$$

$$\sum_{p=0}^{Ub} (h_{i(r+p)} = 1) \leq Ub \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub\} \end{array} \tag{31}$$

$$\sum_{p=0}^{Ub} (h_{i(r+p)} = 0) \leq Ub \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub\} \end{array} \tag{32}$$

$$\sum_{p=0}^{Ub+B} (h_{i(r+p)} = 1) \geq 1 \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - b\} \end{array} \tag{33}$$

$$\sum_{p=0}^{Ub+B} (h_{i(r+p)} = 0) \geq 1 \qquad \begin{array}{l} \forall i \in T, \\ \forall r \in \{1, \ldots, |R| - Ub - b\} \end{array} \tag{34}$$

$$\sum_{i \in T} (h_{ir} = 1) \leq n/2 \qquad \forall r \in R \tag{35}$$

$$\sum_{i \in T} (h_{ir} = 0) \leq n/2 \qquad \forall r \in R \tag{36}$$

$$h_{ir} \in \{0, 1, 2\} \qquad \forall i \in T, \forall r \in R \tag{37}$$

$$v_{ir} \in \{1, \ldots, T\} \qquad \forall i \in T, \forall r \in R^0 \cup \{|R| + 1\} \tag{38}$$

The total distance is calculated by (19). Constraints (20) fix the value of h_{ir} to 2 whenever team i has a bye in round r and they indicate that a bye is spent at the venue of the previous opponent. Constraints (21) (and (22)) limit the number of home (away) games of a team to $n-1$. Whenever a team plays a match in round r, exactly one of the two teams playing that match can play at home, the other plays away. This is enforced by constraints (23). Constraints (24) are used to ensure that team i can only play one home game against every opponent. Constraints (25) - (26) assign the correct venues for every team in every round. All teams should start and end the competition at their home venue, which is ensured by constraints (27) - (28). The number of consecutive home and consecutive away games is limited by respectively constraints (29) and (30). The operation of these constraints is similar to constraints (9)-(10) in the CP-formulation for the RTTP (Section 4.2.1). These constraints, together with constraints (37)-(38) that give the proper limitations to the decision variables, are sufficient for modelling the RTCDMP.

However, the performance of this sufficient model can be enhanced by adding some redundant constraints. These constraints are implemented in a similar way as in the formulation from the previous section and are therefore not extensively covered in detail. Constraints (31) and (32) limit the number of home and away games in Ub+1 rounds to Ub. Constraints (33) and (34) state that at least one home game and one away game should be present in any set of Ub+B+1 consecutive rounds. Finally, constraints (35) and (36) limit the number of home and away games in single round to at most n/2.

Again the variable selection strategy is explicitly listed in the OPL code based on this formulation (Appendix B). The h_{ir} variables are fixed first, since making these decisions will automatically result in fixing the v_{ir} variables.

4.2.3 Discussion on the Modelling Choices

There is no such thing as 'the' IP- or CP-formulation for any given problem, i.e. the proposed formulations are only one way of modelling the problems. Therefore, the modelling choices and their alternatives require a brief explanation. The main discussion point is the choice for CP-modelling over IP-modelling. While both modelling techniques have their advantages, CP offers more room to model problems intuitively. Furthermore, IP-formulated problems tend to become rather complex and overwhelming at first, where the CP-formulation enhances the readability.

Additionally, the complexity of the RTTP results in the fact that an optimal solution, at least for larger instances, will not be found in a reasonable timeframe by using discrete optimization techniques. This is why the main motivation behind developing these formulations is not to solve the problems to optimality. These models are presented to serve as additional benchmarking opportunities for the algorithm proposed in earlier sections. However, in order to provide a benchmark a feasible solution has to be returned first. While they are both very effective for solving combinatorial problems, CP and IP use different strategies to come up with solutions. IP focuses on finding a good but often infeasible solution through linear relaxation, while a branching tree should provide feasibility. CP, on the other hand, tries to maintain feasibility along the search by repetitively extending partial feasible solutions to complete solutions (Hooker, 2002). Since the feasibility of the results is an essential characteristic to provide a benchmark,

a CP-formulation is preferred over an IP-formulation for its purpose in this dissertation.

Another important decision is the choice of the different decision variables of both models. In both models the variables v_{ir} are added since they provide the possibility of modelling the total distance travelled by a team intuitively and efficiently. These variables could probably be omitted in another model formulation, but this would no longer allow to calculate the total distance by simply iterating over all rounds for all teams. Additionally, the implementation of the v_{ir} variables enhance the usability for other problem specifications where byes can only be spent at home or where venue availability plays a crucial role. Finally, Bao (2009) also uses similar decision variables to express the location of any team at any given round in the tournament.

For the RTTP-formulation, the second type of decision variables could also be chosen slightly differently. The x_{ij} variables now represent the round where the home game of team i against team j takes place. However, defining x_{ijr} as a boolean which equals 1 if team i plays at home against team j in round r, could prove helpful for future research on an IP-formulation for the RTTP. For the CP-formulation, the x_{ij} feels like a more clear choice for both the decision variable as for the implementation of several of the constraints. For the RTCDMP, the h_{ir} are defined in a way that they can take three different values. However, an alternative modelling decision could be to only define h_{ir} for rounds where team i actually plays a game. This is already predetermined by the input. Consequently, h_{ir} could be modelled as a boolean which could again be helpful for an IP-formulation. However, the constraints are somewhat less intuitive and comprehensive when h_{ir} is not defined for all rounds.

Since both models are mainly presented for solving the NL-instance class, symmetry breaking in the proposed models is limited. For other instance classes, CIRC and CON, some sort of rotational symmetry exists (Uthus et al., 2012). Rotating the assigned team numbers, i.e. team 1 becomes team 2, team 2 becomes team 3, etc., for these instance classes will have no effect on the total travel distance since the distance between any two consecutive teams is equal. When studying the distance matrix for the NL-instance class, it is clear that this is not the case, therefore no rotational symmetry is present. For the CON instance class, it is even possible to swap the assigned number of any two teams without changing the total distance to be travelled.

A final remark on the modelling choices is the choice for the timetable constrained version of the distance minimisation problem. For a time-constrained instance, this version matches the version where the opponent sequence is used as an input. However, for a time-relaxed problem the timetable contains byes, which differentiates it from the opponent sequence. Another potential problem to be studied could be the one where a sequence of opponents is used as input and where venues need to be assigned and byes need to be added. However, this problem variant lies even further from the focus of this dissertation and it requires a deeper study on the specific problem to develop a CP-formulation. Therefore, this problem variant falls outside the scope of this thesis and interested readers are referred to potential future research on the subject.

5 Experimental Results

5.1 Practical Implementation

The results of the CP models have been obtained using IBM ILOG CPLEX Optimisation Studio (Version 12.10.0.0) on a 2.2 GHz Dual-Core Intel Core i5 processor with 4GB RAM running Windows 10. The RPBSA algorithm was implemented using the Java 1.8.0 programming language in the NetBeans IDE 8.2 environment. All java experiments have been performed on a 2.7 GHz Dual-Core Intel Core i5 processor with 8 GB RAM running iOS. As already mentioned in Section 3.5, it is important to take the competitive nature of the research on timetabling into account when comparing the proposed algorithm with previous research. The use of heuristics is usually proposed whenever exact optimisation is not obtainable within a reasonable timeframe (Gigerenzer, Hertwig, & Pachur, 2011). However, excessive runtimes are omnipresent in the results of previous research on the (R)TTP and since most proposed heuristics use some sort of stochastic decisions, the *Mongolian horde* approach (cf. Section 3.5 or Schaerf and Di Gaspero (2006)) is also something that influences the results in this research field. To avoid distorted results as much as possible, the performances, of both the algorithm as a whole and the different proposed adaptations separately, are compared using mean values of our experimental results. This will allow a fair and objective review of the proposed methods. However, we are also not able to suppress the natural competitive behaviour of human beings. Therefore, Section 5.5 provides an overview of the best solutions that were developed in this dissertation for a range of instances. Solutions that improve the current best known values will also be submitted to be added on the RobinX website for round-robin sports timetabling[14].

5.2 RPBSA

This section discusses the design and results of the RPBSA algorithm. First, the initialization of the parameters is covered in Section 5.2.1. Next, results of the experiments are provided in Section 5.2.2 and the influence of the initial schedule in Section 5.2.3. Section 5.2.4 deals with the evaluation of the repair mechanism and Section 5.2.5 with the dynamic probability distribution of move operator selection. Finally, these experimental results are benchmarked against the results from previous research and from the CP-models proposed in this dissertation (Section 5.2.6).

5.2.1 Parameter Testing

The RPBSA algorithm requires the initialisation of 15 different input parameters. Some of these parameters were already explained in detail and values for those have been set after extensive test runs. The repair multiple τ has been the subject of Section 4.1.5.3 and is set to 0.75. Additionally, the initial values for parameters ω_1 and ω_2 of the dynamic operator selection (Section 4.1.5.2) have also been fixed to, respectively, $(\sqrt{n}/2 + 0.005)\%$ and 0.005%.

[14] http://www.sportscheduling.ugent.be/RobinX/

An optimal combination of the 12 remaining parameters could potentially be obtained by performing a full factorial experiment. However, if every parameter would have only two possible levels, which is not the case, 4096 (2^{12}) additional experiments would have to be performed to cover each combination once and determine all main and interaction effects. In practice, since the parameters have more than two potential levels and a single replication is strongly discouraged due to the stochastic nature of the algorithm, parameter tuning is not done using a full factorial experiment design. Instead, parameter tuning is done one parameter at a time by performing test runs based on the default values presented by Anagnostopoulos et al. (2006) and Van Hentenryck and Vergados (2007). Consequently, interaction effects are not considered in this dissertation. More in depth parameter testing could be an interesting subject for future research. The results of the parameter tuning are summarised in Table 32.

Parameters G, W, R, P and C seem to have a considerable impact on the runtime of the algorithm, but a minor influence on its qualitative outcome. Therefore, these parameters are set in a way that they return an algorithm with a manageable runtime. Appendix C presents the outcome of 3 5-minute test runs of every NLx instance for several parameters. Weight factor θ, cooling factors β and ζ and population parameters X and Y have been set to several values during this test but only marginal gains could be witnessed and are therefore set to their default value as determined by Van Hentenryck and Vergados (2007). The same reasoning is followed for weight factor w, where the default value of 4000 turns out to be the best performing out of the 5 tested values. For the last parameter, T, the differences in outcome noticed between several values were significantly larger, as shown in Table 33. For instances of up to 8 teams, a temperature set to 400 generates better schedules. However, this changes for larger instances, where a temperature of 1000 is performing better. Therefore the value of T is set to 400 for instances of up to 8 teams and 1000 otherwise.

Parameter	Abbreviation	RPBSA
weight	w	4000
temperature	T	$\begin{cases} 400 & \text{if } n \leq 8 \\ 1000 & \text{otherwise} \end{cases}$
cooling factor	β	0.9999
phases of generations	G	10
max stable waves	W	5
population size	X	30
elite runs	Y	10
weight factor	θ	1.04
reheats	R	3
phases	P	50
counter	C	100
cooling factor phase	ζ	0.96
repair multiple	τ	0.75
new best feasible	ω_1	$(\sqrt{n}/2 + 0.005)\%$
new best infeasible	ω_2	0.005%

Table 32: Initial Value Parameters

Temperature	NL8K1	NL8K2	NL8K3	NL10K1	NL10K2	NL10K3	NL12K1	NL12K2	NL12K3
400	40365	41413	39761	67516	69112	66584	130663	127024	125521
700	42886	41471	40788	68796	64202	64143	130120	127986	126932
1000	43032	42254	40548	66541	66201	63370	124714	127974	121613

Table 33: Temperature Parameter Tuning

5.2.2 Experimental Design and Results

Once the parameters are initialized, the RPBSA algorithm is tested together with four variations on the proposed specifications by means of 3 different test runs of 30 minutes for each instance. Additionally, the CP formulation is tested as an additional benchmarking opportunity. The mean of the results are reported in Table 34. A first look at the table learns that optimal solutions are found for the TTP instance and the three RTTP instances with 1,2 and 3 byes for the smallest instance NL4 by all six methods. Optimality of the solutions for NL4 with up to two byes, has been confirmed by the results of the CP formulation.

The first of those six models to be tested is the CP formulation, implemented in IBM ILOG CPLEX Optimization Studio. Only one test run is performed for this approach since this exact method is always performed in the same way for an instance. The results of the CP program are rather competitive but are never the single best performing method. Especially for competitions with 12 or more teams and with 0 or 1 byes, CP seems to be struggling to come up with high-quality solutions. For those instances, all other methods are performing remarkably better. A remark should be made on the increasing performance of CP as the number of byes gets larger, certainly for the larger instances.

In the second row of Table 34, the general algorithm as it is proposed throughout this master's dissertation is tested, and is called RPBSA. The algorithm includes the repair mechanism as presented in Section 4.1.5.3 and the moves are selected using a dynamic distribution as discussed in Section 4.1.5.2. This adapted select move mechanism has an upper limit on the probability an operator is chosen, namely 50% so on average half of the moves should be performed by a different operator. For the time-constrained instances no solution is presented since the repair mechanism is not capable of handling those instances as it makes use of byes to repair the schedules. For four different instances, the RPBSA program has the lowest mean result. For the other instances, RPBSA performs at a reasonable level and is not far off of the results of the best method for this instance.

A third variant is called NO REPAIR and is basically the same as the previous method but without the repair mechanism. Because the repair method is not implemented, results for time-constrained instances can be found. For those TTP problems, NO REPAIR is able to find the optimal solution for 4 and 6 teams, and for 12 teams it delivers the lowest mean result. For RTTP instances, NO REPAIR proves to be the best performing method for only one instance, NL16K1.

The fourth method is called UNIFORM and neglects the dynamic move selection and implements a uniform distribution for all moves at all times. It does use the repair mechanism but only for the time-relaxed instances. This UNIFORM variant can be considered the best performing one since it finds

the lowest mean result for 20 instances out of the presented 28. The resulting mean values are always within 10% of the best solution ever found and for competitions with 4, 6, 8, 12 and 16 teams it returns a schedule that comes within 5% of the best known schedules.

The fifth variant is called MOVES 100% and is almost the same as the RPBSA program except for removing the limit of 50% on the dynamic probability during operator selection. Again, for time-constrained instances the repair mechanism is not implemented. For those instances MOVES 100% is able to present the best performing solutions for 4 different competition sizes. For the time-relaxed instances, this method does never return the lowest mean result.

The method BEST START uses the RPBSA variation but with another initial schedule, the results will be discussed in Section 5.2.3 (cf. infra). The bottom row shows the best known solution found by any method in previous literature. For the first five methods, the method with the lowest mean value for an instance is marked. The BEST START receive a mark when they represent an improvement of the best solution known from previous literature.

5.2.3 Impact Initial Schedule

The schedule that initializes the RPBSA algorithm plays a critical role in the performance of the method. The five discussed methods in Table 34, except for CPLEX, all start from the schedule generated by the 5.875-approximation algorithm proposed by Westphal and Noparlik (2014) as explained in Section 4.1.4. The last method, called BEST START, on the penultimate row of Table 34 uses the RPBSA method, but with another initial schedule. This method starts from the best TTP solution presented on the RobinX website[15] for every instance. This TTP solution is set to a time-relaxed instance in the way presented in Section 3.3.3. The impact of this higher quality starting solution is considerable. For all instances but two, the one with a better starting solution is able to find better results after a 30 minute test run. On top, the previous best known solution is improved for 13 instances by the BEST START method. In order to see the clear impact of the initial schedule, BEST START and RPBSA should be compared. Those 2 methods have these same moves, dynamic selection probability and repair mechanism, only the starting solution differs. Their comparison in Table 34 learns that BEST START is performing better for all instances, which clearly shows the impact of a better starting schedule on runs of 30 minutes. However, the method with the better starting solution does not guarantee a better result. The risk of ending up in a local minimum is quite high, since not a lot of moves will be accepted.

[15] http://www.sportscheduling.ugent.be/RobinX/

Instance	NL4				NL6				NL8				NL10			
	K=0	K=1	K=2	K=3	K=0	K=1	K=2	K=3	K=0	K=1	K=2	K=3	K=0	K=1	K=2	K=3
CPLEX	8276	8160	8160	8044	23916	23188	22796	22707	41686	41690	40211	41433	68124	66601	64980	63512
RPBSA	-	8160	8160	8044	-	23233	22947	22800	-	41268	41404	40580	-	66399	65347	64870
NO REPAIR	8276	8160	8160	8044	23916	23260	23403	22850	39843	43070	42656	41434	62819	68228	67474	66979
UNIFORM	8276	8160	8160	8044	23916	22927	22762	22716	39796	40228	39673	39702	62526	67035	65681	64042
MOVES 100%	8276	8160	8160	8044	23916	23019	22831	22739	39721	41522	40902	40485	63231	68805	67142	64297
BEST START	8276	8160	8160	8044	23916	23019	22557	22720	39721	39255	39148	39255	59436	59144	59021	58833
BEST KNOWN	8276	8160	8160	8044	23916	23124	22557	22557	39721	39128	38761	38670	59436	59425	59373	59582

	NL12				NL14				NL16			
	K=0	K=1	K=2	K=3	K=0	K=1	K=2	K=3	K=0	K=1	K=2	K=3
CPLEX	133744	132041	127451	122860	245229	249096	235825	225234	347338	353851	328996	319062
RPBSA	-	124358	123519	123928	-	238072	233765	231746	-	326541	321712	318679
NO REPAIR	118471	127487	126501	124955	208314	234570	242475	231979	296854	312298	315382	314201
UNIFORM	120544	126350	125949	120584	204531	232990	229596	221064	301032	322939	312214	309169
MOVES 100%	118856	126164	124034	123078	217566	236429	232061	228621	288702	329235	330080	318888
BEST START	115072	114189	113432	113358	207052	206307	203679	202910	287995	284468	281912	281004
BEST KNOWN	110729	117680	119067	116082	188728	209616	209317	205690	261687	307125	300188	297426

Table 34: Mean Experimental Results

CPLEX = CP using CPLEX ILOG
RPBSA = Standard program with repair mechanism and dynamic move selection with a limit of 50% probability of a move being chosen
NO REPAIR = Program without repair mechanism but with dynamic move selection with a limit of 50% probability of a move being chosen
UNIFORM = Program with repair mechanism and a constant uniform distribution for move selection
MOVES 100 = Normal program with repair mechanism and dynamic move selection with a limit of 100% probability of a move being chosen
BEST START = Program starts with the best known solution for the related TTP instance

5.2.4 Repair Mechanism

The results for assessing the performance of the repair mechanism are shown in Figure 10 and the percentages represent the number of infeasible schedules where feasibility is restored divided by the total number of infeasible schedules. For example, the repair probability of NL10K3 is 89.8%, meaning that all generated infeasible schedules with up to 8 violations ($= \lceil \tau n \rceil = \lceil 0.75 * 10 \rceil$) are passed through the repair mechanism. From all those infeasible schedules, 89.8% are returned by the repair mechanism without any violation and consequently, with a restored feasibility. A first conclusion of Figure 10 is that all generated infeasible NL4 schedules with up to 3 violations can be repaired. In fact, only schedules with 2 violations can be passed to the repair mechanism since only the norepeat constraint can be violated. Those violations always come in pairs because two teams play each other in two consecutive rounds and a violation will be reported for each team. A second conclusion is the increasing performance as the number of byes rises. Since the repair mechanism uses byes to repair schedules, more byes provide more opportunities to restore the feasibility of a schedule. For large instances the difference becomes quite extensive, with 3-RTTP instances having a repair probability of over 80% and 1-RTTP instances struggling to get to 25%. The third and last conclusion is the declining performance when the size of the competition increases. First of all, the allowed number of violations is a linear function of the competition size which results in more violations that potentially have to be repaired for larger competitions. Secondly, swapping a match between 2 rounds will cause more teams and matches to be involved for larger instances and therefore reducing the probability of restoring feasibility.

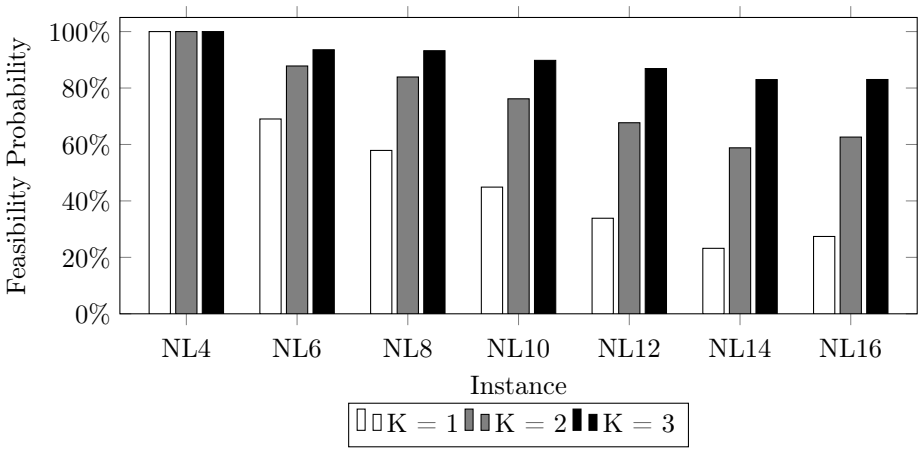


Figure 10: Performance Repair Mechanism

The comparison between an algorithm with and without the repair method is presented in Figure 11b. The squares and crosses represent the mean solution vales after 3 runs of 30 minutes for a program with and without the repair mechanism, respectively. For all instances, except the largest one, the repair method outperforms the other. This confirms the presumption implementing a repair chain is potentially beneficial for the performance of the algorithm.

5.2.5 Dynamic Probability Distribution

To select which move operator to select next, a random variable is generated. In case the distribution is uniform, a random integer between 1 and the number of available moves is generated. The probability of a certain move to be selected is (1/numberOfMoves*100)% in each iteration. However, Section 4.1.5.2 showed that not every move has the same expected performance and the performance is dependent on the selected instance or competition. Therefore a dynamic move selection was proposed with an adapting probability for each operation in each iteration. If a move results in a better feasible or infeasible schedule, its selection probability is increased, whereas the probability of all other moves decreases. For this dynamic move selection, 2 variants exist. One variant implements a limitation on the probability a move can be chosen. In this master's dissertation, 50% was chosen as a benchmark, by doing this in theory every one in two moves should be different. The second variant does not restrict the probability of an operator with the result that the probability of a move operator can converge to 100% and no other operator can be selected.

Figures 11c and 11d show a comparison between those 3 types of methods. Figure 11c compares the uniform move selection method to the dynamic move selection with a limitation of 50%. The triangles represent the uniform distribution and are lower for all instances, except for NL10 and NL12 where both methods have approximately the same gap to the best known solution. For NL14 and NL16, the difference is more than 2% in favor of the uniform distribution. This difference in performance can have several explanations. A first one is that the variety of different moves throughout the program can be effective to find new, potentially better solutions. This reduction of variety by lowering the probability of selection for certain moves can cause the algorithm to get stuck in a local minimum where no move is able to get out. A second explanation can be the one of coincidence. This comparison is based on 3 runs of 30 minutes for both methods and if the right move is selected at the right time, a high-quality schedule can be produced. A third potential explanation can be the extra computational effort that is required to calculate the probability of each move after every iteration. Although these calculations are rather limited, doing it thousands or millions of times might lower the efficiency of the program. The conclusion is that a uniform move selection performs on average slightly better than a dynamic move selection, based on the experimental design in this section.

To assess the necessity of a limitation on the probability of a move, Figure 11d is used. The unlimited dynamic move selection, called Moves 100 and displayed as stars, performs slightly worse than the 50% limited variant. Both methods have the upper hand in 3 of the 6 instances but if the limited variant is better, the difference is larger than for the inverse case. For NL10 and NL16 the difference between both methods is around 2%, with the limited variant being the best performing one. Again the explanation for those differences is not clear but some possible reasons are the same as the previous comparison with the uniform distribution. The overall conclusion of Figure 11c and 11d is that the uniform distribution performs better, followed by the limited dynamic move selection and the unlimited dynamic distribution being the worst performing variant.

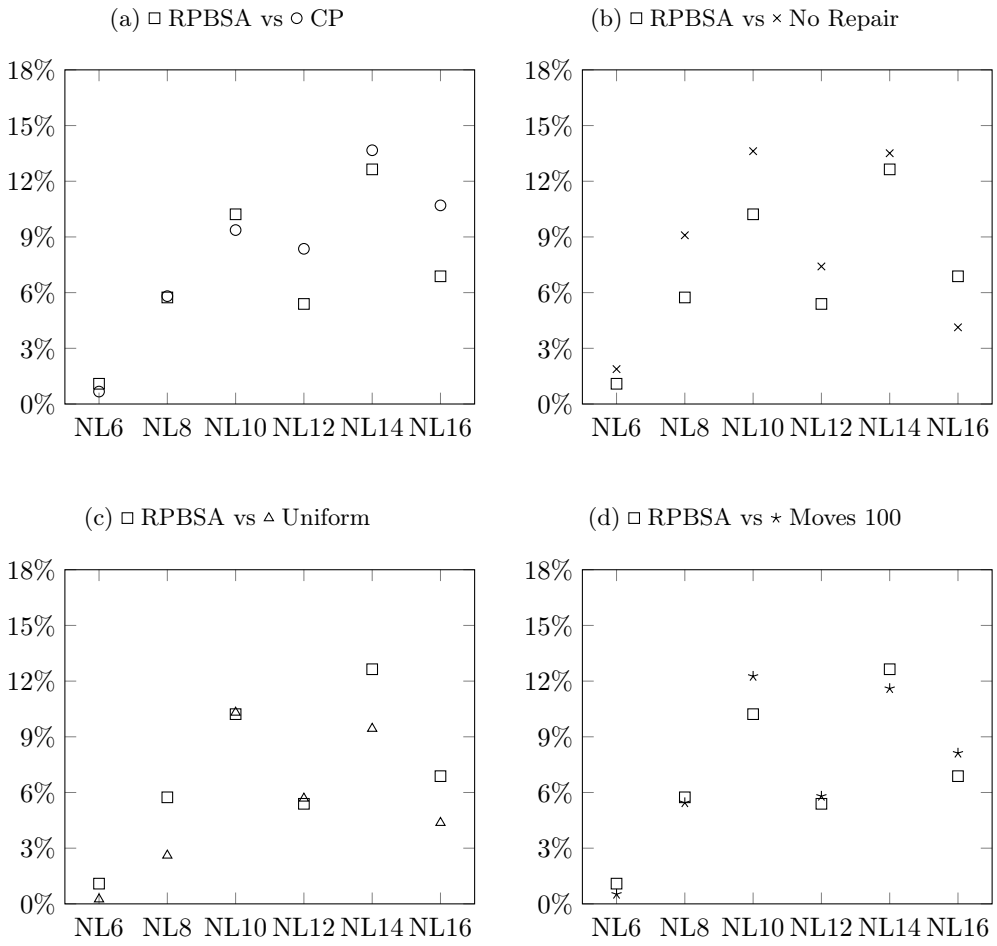


Figure 11: Comparison methods

5.2.6 External Benchmarking

To assess the performance of the constructed heuristics, a comparison is executed with an exact method. The used external benchmark is the CP-formulation presented in 4.2.1. Additionally, the results from this CP-formulation are compared with the results found by Model 22 of Bao (2009). The comparison of the heuristic (RPBSA) and exact method is shown in Figure 11a and a first look learns that the performance of both methods is similar for competitions of up to 10 teams. For the larger instances, the heuristic performs better which could first of all be explained by the initial solution. The heuristic starts from a schedule with a reasonable quality whereas the exact method starts from scratch. Additionally, the experiment design favors the performance of the heuristic. While heuristics are proposed to return high quality solutions in a reasonable timeframe, exact methods are constructed to solve problems to optimality. Given an unlimited runtime, the exact method would always return an optimal solution. However, the experiment mirrors reality by limiting the amount of resources, in this case time, someone has to solve a problem. In realistic scenarios, where these resource limitations apply, the heuristic method clearly performs better.

Where Table 34 contains the mean of the results of 3 test runs of 30 minutes for different methods, Table 35 presents the best solution found for each instance over all methods. The first column shows the objective function value of this best solution. The second and third column represent the gap to the previous best known solution and the method responsible for finding this solution, respectively. The gap is calculated as follows: $\frac{C(S_{bestfound)}-C(S_{previousbest})}{C(S_{previousbest})}$.

Except for method 1, the exact CP-model using CPLEX, all methods start from the initial solution presented in Section 4.1.4. The definition of all methods can be found below Table 35. It should be noted that method 2 (the standard program) is not able to cope with time-constrained instances because of the repair mechanism. A first conclusion is that all methods find the optimal solution for NL4 instances. A second conclusion is the better performance for time-relaxed instances that are a multiple of 4. The gap for NL10 and NL14 instances is at least 5% while this is lower for all other instances, a finding also supported by Westphal and Noparlik (2014). They obtained a lower approximation ratio for instances where n is a multiple of 4 and consequently the initial schedule for those instances will be of higher quality. A third conclusion is that for two instances, an improvement on the current best found solution is returned. For NL6K1 and NL16K1 the best known solution is improved with 1.05 and 0.17 percent, respectively. The new best solutions are 22881 and 306601 found by method 4, using the uniform move selection and method 3, the method without the repair chain. Note, however that the current best NL16K0 solution (261687) can also be used as an NL16K1 solution. A last conclusion is that method 4 finds the best solution for most instances, 22 out of the 28 researched instances. For competitions with 3 byes, this method finds all the best solutions according to Table 35.

Byes	K=0			K=1		
	Value	Gap	Method	Value	Gap	Method
NL4	8276	0.00%	1,3,4,5	8160	0.00%	1,2,3,4,5
NL6	23916	0.00%	1,3,4,5	22881	-1.05%	4
NL8	39721	0.00%	3,4,5	39935	2.06%	4
NL10	62305	4.83%	3	62934	5.90%	4
NL12	117313	5.95%	3	121800	3.50%	4
NL14	201773	6.91%	4	225715	7.68%	4
NL16	288702	10.32%	5	306603	-0.17%	3

Byes	K=2			K=3		
	Value	Gap	Method	Value	Gap	Method
NL4	8160	0.00%	1,2,3,4,5	8044	0.00%	1,2,3,4,5
NL6	22557	0.00%	4	22595	0.17%	4
NL8	39125	0.94%	4	39630	2.48%	4
NL10	64009	7.81%	2	63738	6.98%	4
NL12	122826	3.16%	2	118199	1.82%	4
NL14	226048	7.99%	4	216387	5.20%	4
NL16	306191	2.00%	4	307663	3.44%	4

Table 35: Best Solutions Found

1 = CP using CPLEX ILOG
2 = Standard program with repair mechanism and dynamic move selection with a limit of 50% probability of a move being chosen
3 = Program without repair mechanism but with dynamic move selection with a limit of 50% probability of a move being chosen
4 = Program with repair mechanism and a constant uniform distribution for move selection
5 = Normal program with repair mechanism and dynamic move selection with a limit of 100% probability of a move being chosen

5.3 New Move Operators

5.3.1 Extend Away Trip

The first newly developed move operator is called *ExtendAwayTrip* and basically extends an away trip of length Ub-1 with an additional away game to obtain a trip with a length that matches the upper bound Ub. To assess its performance, the percentage of improved solutions caused by this move is studied. Figure 12 shows this percentage over all NLx instances compared to the average percentage, which would be the result if all operators returned an equal amount of improved schedules. For *ExtendAwayTrip*, the performance is above average for NL4, NL12 and NL14, meaning that this operator is responsible for more improved moves than it is expected to. For the other instances, *ExtendAwayTrip* is relatively close to the average threshold which leads to the conclusion that this move performs well and is beneficial to in the search for an optimal solution.

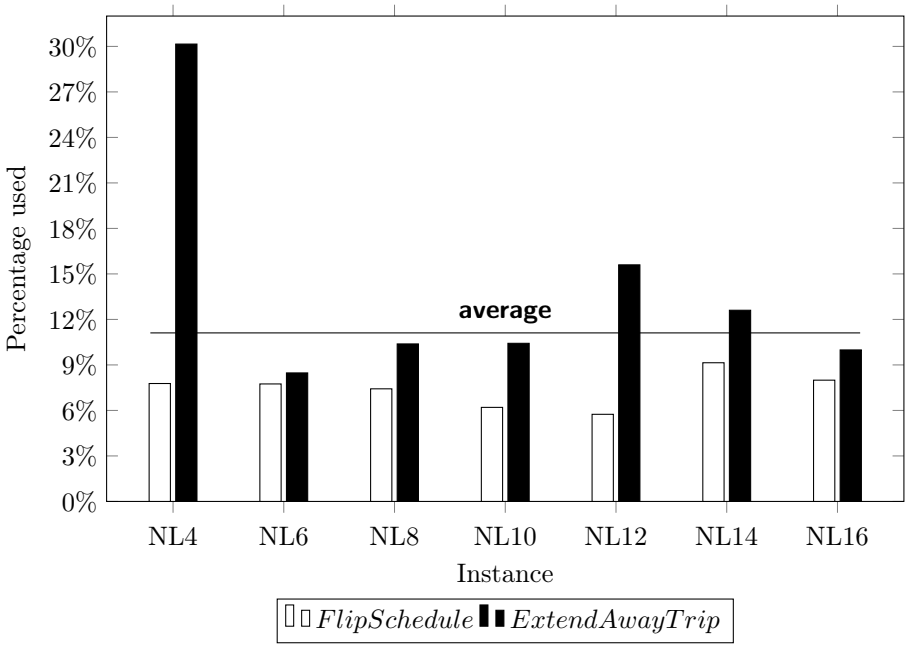


Figure 12: New Move Operator Performance

5.3.2 Flip Schedule

The second developed move is *FlipSchedule* and shuffles the schedule by changing the order of a number of consecutive rounds. Taking a look at Figure 12 learns that this move operator performs below average for all NLx instances. This was to be expected, since this operator normally swaps a lot of entries in the schedule. Therefore, the operator is highly likely to invoke some violations. This might explain the lower performance of around 7% of *FlipSchedule*. Note, however, that this operator was mainly introduced as a diversification strategy.

5.4 RTCDMP

As mentioned before, the CP-model for the RTCDMP can be used to enhance the results of any RTTP-algorithm. Given enough time, the CP-model will always return an optimal home-away assignment for a given opponent schedule and prove its optimality. However, since available runtime is limited, the runs in this experiment are limited to 30 minutes each. The goal of this experiment is to develop the best home-away assignments given the timetables of the solutions in Table 35. Given enough time, this experiment should either prove optimality of the current home-away assignment or it should come up with a better home-away assignment, in terms of objective function value. In practice, this means that the solutions returned by the CP-model for RTCDMP can not be further improved by only adapting the home-away assignment.

The results of this experiment are provided in Table 36. Solutions that improved the RPBSA solution are indicated in light grey. Solutions indicated in dark grey are instances were the solution provided by RPBSA could not be matched within 30 minutes. Only for NL12K1 and NL16K2, a better solution could be provided than the one returned by the RPBSA algorithm. These improvements are also quite minor ($< 1\%$). Furthermore, the optimality of the RPBSA home-away assignments for the timetables up to 8 teams could be proven in less than 20 minutes. Consequently, the results of this experiment clearly show that the solutions provided by the RPBSA leave little to no room for improvement by only adapting the home-away assignment.

		K = 0	K = 1	K = 2	K = 3
NL4	RTCDMP	8276	8160	8160	8044
	RPBSA	8276	8160	8160	8044
NL6	RTCDMP	23916	22881	22557	22595
	RPBSA	23916	22881	22557	22595
NL8	RTCDMP	39721	39935	39125	39630
	RPBSA	39721	39935	39125	39630
NL10	RTCDMP	62305	62934	64009	63738
	RPBSA	62305	62934	64009	63738
NL12	RTCDMP	117313	121649	122826	118199
	RPBSA	117313	121800	122826	118199
NL14	RTCDMP	201773	225715	226048	216387
	RPBSA	201773	225715	226048	216387
NL16	RTCDMP	313614	306603	305830	314797
	RPBSA	288702	306603	306191	307663

Table 36: Results RTCDMP

5.5 Best Experimental Results over Several Instances

Table 37 contains the best solution found over all used methods presented in this master's dissertation for every investigated instance. These solutions are also presented to be uploaded on the RobinX website. Furthermore, these best solutions are compared to the previous best known. Additionally, in Tables 38, 39 and 40, the result of a 15-minute run is presented for all GALAXY, SUPER and CONSTANT instances, respectively. The best known solutions are given as a benchmark but it should be noted that those are usually the result of runs with an excessive runtime. Additionally, it is important to repeat that solutions for the TTP are feasible for the K-RTTP for all $K \geq 0$ (and even, solutions of k_1-RTTP are feasible for k_2-RTTP, for $k_1 \leq k_2$) due to the limited number of byes (Bao & Trick, 2010). However, while solutions with a lower objective function value are sometimes found for instances with less byes (compare for example the solutions for NL8K2 and NL8K3), in accordance with previous literature, solutions are only posted for the instances they have been developed for.

		K = 0	K = 1	K = 2	K = 3
NL4	RPBSA	8276	8160	8160	8044
	best known	8276	8160	8160	8044
NL6	RPBSA	23916	22881	22557	22595
	best known	23916	23124	22557	22557
NL8	RPBSA	39721	39255	39125	39255
	best known	39721	39128	38761	38670
NL10	RPBSA	59436	59144	59021	58833
	best known	59436	59425	59373	59582
NL12	RPBSA	115072	114189	113432	113358
	best known	110729	117680	119067	116082
NL14	RPBSA	201773	206307	203679	202910
	best known	188728	209616	209317	205690
NL16	RPBSA	287995	284468	281912	281004
	best known	261687	307125	300188	297426

Table 37: NL Results

		K = 0	K = 1	K = 2	K = 3
GALAXY4	RPBSA	416	414	413	412
	best known	416	414	413	412
GALAXY6	RPBSA	1365	1346	1323	1322
	best known	1365	1330	1294	1294
GALAXY8	RPBSA	2382	2567	2623	2519
	best known	2373	2298	2261	2250
GALAXY10	RPBSA	5228	5291	5303	4933
	best known	4535	-	-	-
GALAXY12	RPBSA	7982	8659	8520	8700
	best known	7197	-	-	-
GALAXY14	RPBSA	12685	14830	14333	14156
	best known	10918	-	-	-
GALAXY16	RPBSA	12685	14830	14333	14156
	best known	14900	-	-	-
GALAXY18	RPBSA	17571	19234	19080	19003
	best known	20845	-	-	-
GALAXY20	RPBSA	22322	25486	25369	23711
	best known	26289	-	-	-
GALAXY22	RPBSA	30002	35772	34258	35221
	best known	33901	-	-	-
GALAXY24	RPBSA	38985	44228	44083	44187
	best known	44526	-	-	-
GALAXY26	RPBSA	49353	54904	57079	55371
	best known	58968	-	-	-
GALAXY28	RPBSA	69188	75245	76435	75640
	best known	75276	-	-	-
GALAXY30	RPBSA	86803	92432	96080	96115
	best known	95158	-	-	-
GALAXY32	RPBSA	109776	116698	117548	116439
	best known	119665	-	-	-
GALAXY34	RPBSA	134661	147103	143496	148554
	best known	143298	-	-	-
GALAXY36	RPBSA	170895	180629	186464	184100
	best known	169387	-	-	-
GALAXY38	RPBSA	217941	230143	228137	220564
	best known	204980	-	-	-
GALAXY40	RPBSA	252843	264233	266126	265039
	best known	241908	-	-	-

Table 38: GALAXY Results

		K = 0	K = 1	K = 2	K = 3
SUPER4	RPBSA	63405	63334	63263	63192
	best known	63405	63334	63263	63192
SUPER6	RPBSA	130365	128262	127686	127687
	best known	130365	127903	127370	127370
SUPER8	RPBSA	182409	213379	189773	188385
	best known	182409	178115	177406	177258
SUPER10	RPBSA	324380	376487	359465	355627
	best known	316329	-	-	-
SUPER12	RPBSA	479968	546699	502524	500147
	best known	460998	-	-	-
SUPER14	RPBSA	632764	743465	744924	708858
	best known	571632	-	-	-

Table 39: SUPER Results

		K = 0	K = 1	K = 2	K = 3
CON4	RPBSA	17	16	16	16
	best known	17	16	16	16
CON6	RPBSA	43	42	42	42
	best known	43	42	42	42
CON8	RPBSA	81	81	81	80
	best known	80	80	80	80
CON10	RPBSA	129	129	129	129
	best known	124	-	-	-
CON12	RPBSA	185	182	184	185
	best known	181	-	-	-
CON14	RPBSA	269	269	268	265
	best known	252	-	-	-
CON16	RPBSA	340	342	342	341
	best known	327	-	-	-
CON18	RPBSA	423	425	425	425
	best known	417	-	-	-
CON20	RPBSA	549	549	551	548
	best known	520	-	-	-
CON22	RPBSA	648	651	653	650
	best known	626	-	-	-
CON24	RPBSA	756	761	764	762
	best known	749	-	-	-

Table 40: CON Results

6 General Conclusion and Future Work

In this section, some general conclusions from this dissertation are listed. First of all, it is important to note that the results presented in Section 5 only serve as indicators for the general conclusions. Due to limited available runtime, limited CPU power, the stochastic nature of the proposed algorithm and the decision to present results for a broad range of instances, only a select number of experiments could be performed. Therefore, no hard proof could be provided for the different conclusions. However, we feel that valuable conclusions can still be drawn based on the indications presented in the results.

A first conclusion that can be made is the fact that a heuristic approach is preferred over an exact optimisation approach for the RTTP, if runtime is limited. This is indicated by the fact that the results provided by the heuristic PBSA algorithm clearly outperform those of the CP model on 30 minute runs. Note, however, that while 30 minutes usually is considered as a long runtime for heuristics, this is not the case for heuristics for the (R)TTP where a runtime of more than one day is not unusual (Anagnostopoulos et al., 2006).

A second set of conclusions covers the proposed adaptions to the TTSA algorithm. The results give a strong indication that the implementation of a repair mechanism probably enhances the performance of the RPBSA algorithm. Additionally, the two new move operators ($ExtendAwayTrip$ and $Flipschedule$) both have the potential to deliver (better) feasible schedules, which indicates that implementing these extra operators is helpful in the search for an optimal solution. It is also important to notice the improved short term results obtained by starting the algorithm from a schedule with a lower objective function value. Almost all best results for the NL-instance class have been found by using the best TTP solution available on the RobinX website as an initial schedule. While the short term results for this approach seem promising, it is important to keep mind that the initial schedules might represent local minima in the objective function. Consequently, the search for an optimal solution might be affected negatively by using an initial schedule that is too good.

Finally, the results from Section 5 do not indicate any added value in dynamically selecting the applied move operators. While it seems intuitive that favouring the best-performing move operators would return better results, this claim can not be confirmed by the experiments. A potential explanation for this is the fact that the diversification provided by having more variation in the move operators is more valuable for the search than initially estimated.

This dissertation contributes to the research field of sports timetabling, and in particular time-relaxed timetabling, by proposing a heuristic approach and two exact optimisation methods. Additionally, this dissertation provides a structured overview of past literature on the subject. However, some aspects of sports timetabling problems, while interesting, fall outside the scope of this dissertation and remain to be researched in future work.

First of all, it could be interesting to implement the rationale behind some of the proposed adaptations to a time-constrained problem. Since the TTP is an extensively covered and competitive research subject, a 'quick-and-dirty' implementation of these innovative adaptations might not result in a correct representation of their true potential. Therefore, the conscious choice was made to mainly restrict this dissertation to time-relaxed timetabling problems

and leave the extension to TTP for future research. The targeted search for away trips of suboptimal length, used in the move operators $GroupingAwayUsingByes$, proposed by Pérez-Cáceres and Riff (2015), and $ExtendAwayTrip$, proposed in this dissertation, could for example serve as a potential improvement on the existing set of move operators for the TTP. Naturally, the absence of byes would render this task significantly more challenging. A second proposal for future research is the implementation of a repair mechanism for the TTP. Again the absence of byes can make it harder to implement a similar mechanism in a time-constrained environment.

It could also be interesting to try to step away from the theoretical problem formulation and start moving closer to real-life scenarios. The existing models can potentially be extended to take availability constraints and preferences of the teams into account. It might also be interesting to adapt the models in a way that they return more equitable schedules. Additionally, the core of the objective function, i.e. travel distance, could be replaced by a more economical value that also takes e.g. travel and accommodation costs into account. The distance minimisation problem could than be turned into a cost minimisation problem. Finally, it might be interesting to study the practical usability of an 'optimal' schedule. While this schedule might optimise the total distance to be travelled, it might not be desirable to repeat the same schedule year after year, for various reasons.

References

Anagnostopoulos, A., Michel, L., Van Hentenryck, P., & Vergados, Y. (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, *9*(2), 177–193.

Bao, R. (2009). Time relaxed round robin tournament and the NBA scheduling problem. *ETD Archive*, *Paper 25*.

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

Bean, J. C., & Birge, J. R. (1980). Reducing travelling costs and player fatigue in the national basketball association. *Interfaces*, *10*(3), 98–102.

Biajoli, F. L., & Lorena, L. A. N. (2007). Clustering search approach for the traveling tournament problem. In *Mexican international conference on artificial intelligence* (p. 83-93).

Brandão, F., & Pedroso, J. P. (2014). A complete search method for the relaxed traveling tournament problem. *EURO Journal on Computational Optimization*, *2*(1-2), 77–86.

Buraimo, B., Forrest, D., & Simmons, R. (2009). Insights for clubs from modelling match attendance in football. *Journal of the Operational Research Society*, *60*(2), 147–155.

Chen, P.-C., Kendall, G., & Vanden Berghe, G. (2007). An ant based hyper-heuristic for the travelling tournament problem. In *2007 IEEE Symposium on Computational Intelligence in Scheduling* (p. 19-26).

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

Della Croce, F., & Oliveri, D. (2006). Scheduling the Italian football league: An ILP-based approach. *Computers & Operations Research*, *33*(7), 1963–1974.

Di Gaspero, L., & Schaerf, A. (2007). A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics*, *13*(2), 189-207.

Drexl, A., & Knust, S. (2007). Sports league scheduling: graph-and resource-based models. *Omega*, *35*(5), 465–471.

Duffield, R., & Fowler, P. M. (2017). Domestic and international travel: implications for performance and recovery in team-sport athletes. *Sport, recovery, and performance: Interdisciplinary insights*.

Du Preez, M., & Lambert, M. I. (2007). Travel fatigue and home ground advantage in South African Super 12 rugby teams. *South African Journal of Sports Medicine*, *19*(1), 20–22.

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

Durán, G., Guajardo, M., & Sauré, D. (2017). Scheduling the South American Qualifiers to the 2018 FIFA World Cup by integer programming. *European Journal of Operational Research*, *262*(3), 1109–1115.

Easton, K., Nemhauser, G., & Trick, M. (2001). The traveling tournament problem description and benchmarks. In *International Conference on Principles and Practice of Constraint Programming* (p. 580-584).

Easton, K., Nemhauser, G., & Trick, M. (2002). Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In *International conference on the practice and theory of automated timetabling* (pp. 100–109).

Eiben, A., & Ruttkay, Z. (1997). *Constraint satisfaction problems.*

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A Guide to the Theory of NP-Completeness* (Vol. 174). W.H. Freeman & Co.

Gigerenzer, G. E., Hertwig, R. E., & Pachur, T. E. (2011). *Heuristics: The foundations of adaptive behavior.* Oxford University Press.

Glover, F. (1992). New ejection chain and alternating path methods for traveling salesman problems. In *Computer science and operations research* (p. 491-509). Elsevier.

Goossens, D. R. (2017). Optimization in sports league scheduling: experiences from the Belgian Pro League soccer. In *International conference on operations research and enterprise systems* (pp. 3–19).

Goossens, D. R., & Spieksma, F. C. (2012). Soccer schedules in Europe: an overview. *Journal of scheduling*, *15*(5), 641–651.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, *45*(7), 733-750.

Hooker, J. N. (2002). Logic, optimization, and constraint programming. *INFORMS Journal on Computing*, *14*(4), 295–321.

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

Kendall, G., & Westphal, S. (2013). Sports Scheduling: Minimizing Travel for English Football Supporters. In *Automated scheduling and planning* (pp. 61–90). Springer.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680.

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

Langford, G. (2010). *An improved neighbourhood for the traveling tournament problem.*

Lim, A., Rodrigues, B., & Zhang, X. (2006). A simulated annealing and hill-climbing algorithm for the traveling tournament problem. *European Journal of Operational Research*, *174*(3), 1459-1478.

Mann, D. L., Dehghansai, N., & Baker, J. (2017). Searching for the elusive gift: advances in talent identification in sport. *Current opinion in psychology*, *16*, 128–133.

Misir, M., Wauters, T., Verbeeck, K., & Vanden Berghe, G. (2009). A new learning hyper-heuristic for the traveling tournament problem. In *the 8th Metaheuristic International Conference.*

Miyashiro, R., & Matsui, T. (2005). A polynomial-time algorithm to find an equitable home–away assignment. *Operations Research Letters*, *33*(3), 235–241.

Myers, B. R. (2012). A proposed decision rule for the timing of soccer substitutions. *Journal of Quantitative Analysis in Sports*, *8*(1).

Norman, J. M., & Clarke, S. R. (2010). Optimal batting orders in cricket. *Journal of the Operational Research Society*, *61*(6), 980–986.

Padulo, J., Attene, G., Migliaccio, G. M., Cuzzolin, F., Vando, S., & Ardigò, L. P. (2015). Metabolic optimisation of the basketball free throw. *Journal of sports sciences*, *33*(14), 1454–1458.

Pearl, J. (1984). *Intelligent search strategies for computer problem solving.* Addision Wesley.

Peets, H., Müller-Marein, J., & Sommer, T. (1960). *Sport, die wichtigste Nebensache der Welt.* Schünemann.

Pérez-Cáceres, L., & Riff, M. C. (2015). Solving scheduling tournament problems using a new version of clonalg. *Connection Science*, *27*(1), 5-21.

Pisinger, D., & Ropke, S. (2010). Large neighborhood search. In *Handbook of metaheuristics* (pp. 399–419). Springer.

Pollard, R., Prieto, J., & Gómez, M.-Á. (2017). Global differences in home advantage by country, sport and sex. *International Journal of Performance Analysis in Sport*, *17*(4), 586–599.

Post, G., & Woeginger, G. J. (2006). Sports tournaments, home–away assignments, and the break minimization problem. *Discrete Optimization*, *3*(2), 165–173.

Rasmussen, R. V., & Trick, M. A. (2006). The timetable constrained distance minimization problem. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming* (pp. 167–181).

Rasmussen, R. V., & Trick, M. A. (2008). Round robin scheduling - a survey. *European Journal of Operational Research*, *188*(3), 617–636.

Régin, J.-C. (2001). Minimization of the number of breaks in sports scheduling problems using constraint programming. *DIMACS series in discrete mathematics and theoretical computer science*, *57*, 115–130.

Ribeiro, C. C., & Urrutia, S. (2007). Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, *179*(3), 775-787.

Rogulj, N., Papić, V., & Čavala, M. (2009). Evaluation models of some morphological characteristcs for talent scouting in sport. *Collegium antropologicum*, *33*(1), 105–110.

Russell, R. A., & Leung, J. M. (1994). Devising a cost effective schedule for a baseball league. *Operations Research*, *42*(4), 614–625.

Schaerf, A., & Di Gaspero, L. (2006). Measurability and reproducibility in timetabling research: State-of-the-art and discussion. In *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling* (pp. 53–62).

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

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

Suzuka, A., Miyashiro, R., Yoshise, A., & Matsui, T. (2007). The home–away assignment problems and break minimization/maximization problems in sports scheduling. *Pacific Journal of Optimization*, *3*, 113–33.

Thielen, C., & Westphal, S. (2010). Approximating the traveling tournament problem with maximum tour length 2. In *International Symposium on Algorithms and Computation* (pp. 303–314).

Thielen, C., & Westphal, S. (2011). Complexity of the traveling tournament problem. *Theoretical Computer Science*, *412*(4-5), 345–351.

Toffolo, T. A., Christiaens, J., Spieksma, F. C., & Vanden Berghe, G. (2019). The sport teams grouping problem. *Annals of Operations Research*, *275*(1), 223–243.

Trick, M. A. (2000). A schedule-then-break approach to sports timetabling. In *International conference on the practice and theory of automated timetabling* (pp. 242–253).

Trick, M. A., Yildiz, H., & Yunes, T. (2012). Scheduling major league baseball umpires and the traveling umpire problem. *Interfaces*, *42*(3), 232–244.

Urrutia, S., & Ribeiro, C. C. (2004). Minimizing travels by maximizing breaks in round robin tournament schedules. *Electronic Notes in Discrete Mathematics*, *18*, 227–233.

Uthus, D. C., Riddle, P. J., & Guesgen, H. W. (2009a). An ant colony optimization approach to the traveling tournament problem. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (p. 81-88).

Uthus, D. C., Riddle, P. J., & Guesgen, H. W. (2009b). Dfs* and the traveling tournament problem. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (p. 279-293).

Uthus, D. C., Riddle, P. J., & Guesgen, H. W. (2012). Solving the traveling tournament problem with iterative-deepening A*. *Journal of Scheduling*, *15*(5), 601-614.

Van Bulck, D., & Goossens, D. (2020). Handling fairness issues in time-relaxed tournaments with availability constraints. *Computers & Operations Research*, *115*.

Van Bulck, D., Goossens, D., Schönberger, J., & Guajardo, M. (2020). Robinx: A three-field classification and unified data format for round-robin sports timetabling. *European Journal of Operational Research*, *280*(2), 568–580.

Van Bulck, D., Goossens, D. R., & Spieksma, F. C. (2019). Scheduling a non-professional indoor football league: a tabu search based approach. *Annals of Operations Research*, *275*(2), 715-730.

Van Hentenryck, P., & Vergados, Y. (2007). Population-based simulated annealing for traveling tournaments. In *Proceedings of the National Conference on Artificial Intelligence* (Vol. 22, p. 267).

Westphal, S., & Noparlik, K. (2014). A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, *218*(1), 347–360.

Wloch, K., & Bentley, P. J. (2004). Optimising the performance of a Formula one car using a genetic algorithm. In *International Conference on Parallel Problem Solving from Nature* (pp. 702–711).

Appendices

A OPL Code CP- formulation RTTP

```
/*********************************************
 * OPL 12.10.0.0 Model
 * Authors: Alexander De Munster, Bram D'haenens
 *********************************************/
using CP;
int nbTeams = ...;
range teams = 1..nbTeams;
int B = ...;
range byes = 0..B;
int nbRounds = 2*(nbTeams-1)+B;
range rounds = 1..nbRounds;
int Ub = ...;
int D[teams,teams]=...;
dvar int x[teams][teams] in 0..nbRounds;
dvar int v[1..nbTeams][0..nbRounds+1] in 1..nbTeams;
execute{
    writeln("Defining search strategy and setting parameters");
    cp.param.TimeLimit = 30*60;
    var f=cp.factory;
    var phase1 = f.searchPhase(x, f.selectSmallest(f.domainSize()),
                                                    f.selectLargest(f.value()));
    var phase2 = f.searchPhase(v, f.selectSmallest(f.domainSize()),
                                                    f.selectLargest(f.value()));
    cp.setSearchPhases(phase1, phase2);
    cp.param.PresolveLevel=6;
    cp.param.Workers = 4;
    cp.param.AllDiffInferenceLevel = 6;
    cp.param.CountInferenceLevel = 6;
}
minimize sum(i in teams,r in 0..nbRounds)D[v[i][r],v[i][r+1]];
subject to
{
    //One game per day
    forall(i in teams)
        allDifferent(append(all (j in teams: j!=i) x[i][j], all (j in teams) x[j][i]));
    //Not against self
    forall(i in teams)
        x[i][i] == 0;
    //Set venues
    forall(i,j in teams, r in rounds)
        x[i][j] == r => v[i][r] == i && v[j][r] == i;
    //Start and finish at home
    forall(i in teams){
        v[i][0] == i;
        v[i][nbRounds+1] == i;
    }
    //Stay at the same venue during a bye
    forall(i in teams, r in rounds)
        count(append(all(j in teams)x[i][j], all(j in teams)x[j][i]), r) == 0
                                                    => v[i][r] == v[i][r-1];
    //No repeat constraints
    forall(i,j in teams: i<j)
        abs(x[i][j] - x[j][i]) > 1;
```

```
54      //At most constraints home games
55      forall (b in byes,r in 1..nbRounds-Ub+b, i in teams)
56          sum(n in 0..Ub+b)count(all(j in teams)x[i][j], r+n)== 0
57                      => sum(n in 0..Ub+b)count(all(j in teams)x[j][i], r+n) <= Ub;
58      //At most constraints away games
59      forall (b in byes,r in 1..nbRounds-Ub+b, i,j in teams)
60          sum(n in 0..Ub+b)count(all(j in teams)x[i][j], r+n)== 0
61                      => sum(n in 0..Ub+b)count(all(j in teams)x[j][i], r+n) <= Ub;
62      //Redundant Constraints
63      forall (r in 1..nbRounds-Ub-B, i in teams) {
64          sum(p in 0..Ub+B) count(all(j in teams) x[i][j], r+p) >= 1;
65          sum(p in 0..Ub+B) count(all(j in teams) x[j][i], r+p) >= 1;
66      }
67      forall(i in teams, k in 1..nbTeams-1){
68          sum(r in 1..nbRounds-k*(Ub+1)) count(all(j in teams) x[i][j], r) >= nbTeams-1-k*Ub;
69          sum(r in 1..nbRounds-k*(Ub+1)) count(all(j in teams) x[j][i], r) >= nbTeams-1-k*Ub;
70      }
71      forall(r in rounds){
72          count(all(i,j in teams: i!=j) x[i][j], r) <= nbTeams/2;
73      }
74      //Symmetry breaking
75      x[1][2]>x[2][1];
76  }
```

B OPL Code CP-formulation RTCDMP

```
/*********************************************
 * OPL 12.10.0.0 Model
 * Author: Alexander De Munster, Bram D'haenens
 *********************************************/
using CP;

int nbTeams = ...;
range teams = 1..nbTeams;
int B = ...;
int nbRounds = 2*(nbTeams-1)+B;
range rounds = 1..nbRounds;
int Ub = ...;
int D[teams,teams]=...;
int TT[teams][0..nbRounds+1]=...;
dvar int h[0..nbTeams][rounds]in 0..2;
dvar int v[teams][0..nbRounds+1] in 1..nbTeams;
execute{
    writeln("Defining search strategy and setting parameters");
    cp.param.TimeLimit = 30*60;
    var f=cp.factory;
    var phase1 = f.searchPhase(h, f.selectSmallest(f.domainSize()),
                                                    f.selectLargest(f.value()));
    var phase2 = f.searchPhase(v, f.selectSmallest(f.domainSize()),
                                                    f.selectLargest(f.value()));
    cp.setSearchPhases(phase1, phase2);
    cp.param.PresolveLevel=6;
    cp.param.Workers = 4;
}
minimize sum(i in teams, r in 0..nbRounds)D[v[i][r],v[i][r+1]];;
subject to
{
    //Handle occurence of byes
    forall(i in 1..nbTeams, r in 1..nbRounds)
        TT[i][r] == 0 => h[i][r]==2 && v[i][r] == v[i][r-1];
    //Limit number of home/away games to n-1
    forall(i in teams)
    {
        count(all(r in rounds)h[i][r],1)== nbTeams-1;
        count(all(r in rounds)h[i][r],0)== nbTeams-1;
    }
    //For every match one team plays at home and one plays away
    forall(i in teams, r in rounds)
        (TT[i][r]!= 0) => h[i][r] + h[TT[i][r]][r]==1;
    //Play every team exactly once at home
    forall(i in 1..nbTeams, r1,r2 in 1..nbRounds:r1!=r2)
        (TT[i][r1] == TT[i][r2] && TT[i][r1] != 0)
                => h[i][r1]+h[i][r2]==1;
    //Set venue of the games
    forall(i in 1..nbTeams, r in 1..nbRounds)
    {
        h[i][r] == 1 => v[i][r] == i;
        h[i][r] == 0 => v[i][r] == TT[i][r];
    }
```

```
54      //Start and end at the home venue
55      forall(i in 1..nbTeams)
56      {
57            v[i][0] == i;
58            v[i][nbRounds+1] == i;
59      }
60      //Atmost constraint away games
61      forall(b in 0..B, r in 1..(nbRounds-Ub-b),i in 1..nbTeams)
62        sum(a in 0..Ub+b)(h[i][r+a]== 0) == 0
63                           => sum(a in 0..Ub+b)(h[i][r+a]==1) <= Ub;
64      //Atmost constraint home games
65      forall(b in 0..B, r in 1..(nbRounds-Ub-b),i in 1..nbTeams)
66        sum(a in 0..Ub+b)(h[i][r+a]== 1) == 0
67                           => sum(a in 0..Ub+b)(h[i][r+a]==0) <= Ub;
68      //Redundant constraints
69      forall (r in 1..nbRounds-Ub, i in teams) {
70          count(all(p in 0..Ub) h[i][r+p], 0) <= Ub;
71          count(all(p in 0..Ub) h[i][r+p], 1) <= Ub;
72      }
73      forall (r in 1..nbRounds-Ub-B, i in teams) {
74          count(all(p in 0..Ub+B) h[i][r+p], 0) >= 1;
75          count(all(p in 0..Ub+B) h[i][r+p], 1) >= 1;
76      }
77      forall(r in rounds){
78          count(all(i in teams) h[i][r], 1) <= nbTeams/2;
79          count(all(i in teams) h[i][r], 0) <= nbTeams/2;
80      }
81  }
```

C Results Parameter Testing

The results of the following parameter tests are presented as a percentage which is calculated as follows: $\frac{C(S_{new})-C(S_{bestknown})}{C(S_{bestknown})}$

w	NL4	NL6	NL8	NL10	NL12	NL14	NL16	Average
500	0.00%	2.39%	9.40%	14.56%	10.73%	21.11%	11.35%	9.93%
1000	0.00%	2.05%	7.46%	14.95%	7.76%	18.54%	10.65%	8.77%
2500	0.00%	2.21%	8.88%	14.83%	8.59%	19.89%	10.86%	9.32%
4000	0.00%	1.43%	7.65%	14.78%	6.26%	16.24%	12.03%	8.34%
10000	0.00%	1.93%	10.05%	15.20%	6.85%	19.12%	13.78%	9.56%

β	NL4	NL6	NL8	NL10	NL12	NL14	NL16	Average
0.9	0.00%	2.15%	9.72%	14.61%	7.02%	16.52%	9.46%	8.50%
0.95	0.00%	2.05%	5.90%	13.87%	8.09%	19.37%	13.60%	8.98%
0.99	0.00%	2.69%	9.24%	13.63%	7.59%	19.82%	10.72%	9.10%
0.9999	0.00%	1.40%	7.87%	14.40%	9.41%	16.78%	13.20%	9.01%
0.999999	0.00%	2.06%	7.45%	14.62%	6.74%	18.67%	11.94%	8.78%

θ	NL4	NL6	NL8	NL10	NL12	NL14	NL16	Average
1.01	0.00%	1.85%	9.52%	13.27%	8.24%	18.07%	9.88%	8.69%
1.025	0.00%	2.67%	7.35%	13.31%	7.87%	17.30%	12.12%	8.66%
1.04	0.00%	1.68%	8.48%	14.25%	7.47%	17.20%	11.55%	8.66%
1.05	0.00%	2.73%	8.45%	14.33%	7.51%	16.44%	11.33%	8.69%
1.1	0.00%	1.69%	6.69%	13.70%	7.95%	18.92%	10.85%	8.54%

ζ	NL4	NL6	NL8	NL10	NL12	NL14	NL16	Average
0.9	0.00%	1.09%	8.64%	16.48%	7.55%	16.57%	11.77%	8.87%
0.95	0.00%	2.25%	6.17%	13.98%	9.34%	14.75%	12.41%	8.42%
0.96	0.00%	2.85%	8.51%	15.99%	7.23%	19.81%	11.66%	9.44%
0.975	0.00%	1.52%	7.72%	13.66%	9.63%	18.53%	11.53%	8.94%
0.99	0.00%	1.82%	7.06%	16.51%	7.07%	18.92%	11.65%	9.00%

X & Y	NL4	NL6	NL8	NL10	NL12	NL14	NL16	Average	μ_{pop}	μ_g
X = 10, Y = 3	0.00%	1.45%	7.67%	14.61%	6.42%	16.54%	11.97%	8.38%	33.38	3.33
X = 30, Y = 10	0.00%	1.82%	7.76%	13.98%	8.16%	19.63%	11.75%	9.01%	15.95	2.10
X = 30, Y = 20	0.00%	2.34%	7.47%	15.95%	8.22%	18.84%	10.63%	9.07%	15.38	2.00
X = 50, Y = 15	0.00%	1.83%	7.15%	14.09%	8.17%	21.22%	12.85%	9.33%	11.90	1.67
X = 80, Y = 30	0.00%	2.04%	8.71%	15.35%	7.52%	21.14%	14.88%	9.95%	8.71	1.57

μ_{pop} = Average number of populations created
μ_g = Average number of generations created, with a maximum of $G = 5$ (default $G = 10$)