University of Bergen
Department of Informatics

Balancing the Game: Comparative Analysis of Single Heuristics and Adaptive Heuristic Approaches for Sports Scheduling Problem

Author: Seyed Erfan Alesahebfosoul

Supervisor: Ahmad Hemmati

UNIVERSITETET I BERGEN

Det matematisk-naturvitenskapelige fakultet

September, 2023

Abstract

Sport timetabling problems are Combinatorial Optimization problems which involve the creation of schedules that determine when and where teams compete against each other. One specific type of sports scheduling, the double round-robin (2RR) tournament, mandates that each team faces every other team twice, once at their home venue and once at the opponent's. Despite the relatively small number of teams involved, the sheer volume of potential scheduling combinations has spurred researchers to employ various techniques to find efficient solutions for sports scheduling problems.

In this thesis, we present a comparative analysis of single and adaptive heuristics designed to efficiently solve sports scheduling problems. Specifically, our focus is on constructing time-constrained double round-robin tournaments involving 16 to 20 teams, while adhering to hard constraints and minimizing penalties for soft constraints violations. The computational results demonstrate that our adaptive heuristic approach not only successfully finds feasible solutions for the majority of instances but also outperforms the single heuristics examined in this study.

Acknowledgements

I would like to extend my heartfelt gratitude to my dedicated supervisor, Ahmad Hemmati, for his guidance and support throughout my master's studies and the completion of this thesis. I am profoundly thankful to my parents for their constant encouragement, especially my father, who consistently reminded me of the importance of hard work during our weekly conversations.

Lastly, I extend a massive thank you to my incredible wife, Parisa. Her strong support has been the cornerstone of my journey, and without her, I would not have been able to embark on and successfully conclude this academic endeavor.

Seyed Erfan Alesahebfosoul
Saturday 30th September, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Scheduling sports events is a complex task that has caught the attention of a diverse group of researchers from different fields such as operations research, scheduling theory, constraint programming, graph theory, combinatorial optimization, and applied mathematics[5]. These challenges involve creating schedules that specify when, where, and which teams will compete.

In the context of sports scheduling, various constraints and problem types exist, often depending on the specific sport and league. These constraints can encompass venue availability, team preferences, travel constraints, and fairness considerations. The diverse nature of sports timetabling problems makes it difficult to pinpoint the specific papers relevant to a particular problem in this field. Additionally, the absence of a standard data format means that problem examples and their solutions are seldom shared. As a result, it is challenging to evaluate how well algorithms perform because they are typically tested on only a few specific examples. To address these challenges, Van Bulck et al. collected and categorized various problems from the past five decades presented in the literature in the RobinX project [2]. The instances used in this study originate from the problems gathered in the RobinX project.

One might question why it is challenging to schedule a competition with a small number of teams and whether it is possible to list all potential schedules. To illustrate the enormity of the solution space, we can turn to the research of Van Bulck [17]. He demonstrates that when we disregard whether a game is played at home or away and also the order of weekly schedules, scheduling a competition for a group of four teams results in just one possible combination. However, this number skyrockets to 6,240 for six teams and reaches a staggering 252,282,619,805,368,320 for 12 teams, which can indicate the complexity of solving the problem and the necessity of efficient and effective solution methods.

As a result, researchers have delved into these problems, often deploying a variety of optimization techniques to find effective solutions. These techniques encompass exact methods such as integer programming (IP) and constraint programming, as well as heuristic approaches including metaheuristics, matheuristics, and hybrid methodologies. Integer programming provides exact solutions but can be computationally demanding, while heuristic methods offer quicker but approximate solutions. Because sports scheduling is inherently complicated, it requires a range of strategies to meet the rules and goals set by sports leagues and organizations effectively. The choice of optimization method depends on the problem's complexity and the desired quality of the solution. Researchers continually explore new techniques and adapt existing ones to tackle evolving challenges in sports scheduling.

This thesis focuses on automating the schedule generation process for specific instances from the RobinX project, which involves both hard and soft constraints categorized into capacity, game, break, fairness, and separation constraints. Hard constraints must not be violated, while soft constraints result in penalties if violated. The goal is to create schedules that strictly adhere to hard constraints while minimizing the penalty for soft constraint violations. To achieve this, a matheuristic approach is employed, consisting of a mathematical model and a two-phase strategy. The first phase aims to find a feasible initial solution using various methods, while the second phase involves a detailed comparison of different heuristics, including an adaptive heuristic developed in this study.

This thesis has the following structured format. Section 2 lays the foundation by providing essential background information, including problem context, solution methods, key terminology, and relevant research. In Section 3, we delve into a comprehensive explanation of the problem's constraints and offer an overview of the instances considered. Section 4 is dedicated to the mathematical model and provides a detailed overview of our problem and constraints. In section 5, we present our solution approach, which encompasses single heuristics and our adaptive heuristic method. Our experimental results are presented in Section 6, and the thesis concludes and gives further research avenues in Section 7.

Chapter 2

Background

2.1 Combinatorial Optimization

Combinatorial optimization is a field of study within mathematics, computer science, and operations research that deals with finding the best possible solution from a finite set of possible options.[13] The main characteristic of combinatorial optimization problems is that they involve discrete decision variables and often have a large number of possible solutions. These problems are encountered in various real-world scenarios where choices or resource allocations need to be made to optimize a certain objective.

In combinatorial optimization, the goal is to find the optimal combination of elements from a given set in order to maximize or minimize a certain objective function. The *combinatorial* aspect arises because the solutions are formed by combining discrete elements in various ways, and the challenge lies in exploring the vast solution space to find the best configuration. Therefore, researchers have developed various algorithms that can find good solutions quickly, or approximate the optimal solution within some error bound. More details regarding solution methods are provided in section 2.3.

2.2 Sport Scheduling

A sports scheduling problem is a type of optimization problem that involves finding the best way to organize and plan a sports competition, such as a tournament or a league. In other words, sports scheduling problems can be regarded as combinatorial optimization challenges, involving the creation of a schedule that specifies the opponents, venues, and timing of all teams' matches. The primary aim is to meet a range of constraints and goals, including minimizing travel distances, ensuring a fair distribution of home and away games, avoiding scheduling conflicts with venues or television broadcasts, and ensuring an appealing and balanced competition.

The complexity and difficulty of a sports scheduling problem can vary significantly based on factors such as the number of participants, competition format, specific regulations, and the preferences of those involved. In addition to its practical significance, this problem is typically categorized as NP-Hard in the majority of instances, making it infeasible to manually construct high-quality sports schedules [12, 1]. Therefore, mathematical and computational techniques are often used to model and solve sports scheduling problems.

There exists a wide array of sports scheduling problems, depending on the characteristics of the sport and the competition. Some common examples are:

- Round-robin tournaments: A round-robin tournament is a type of tournament in which each participant plays every other participant the same number of times, usually once or twice. The winner of the tournament is the one who has the best performance, measured by the number of wins, points, or other criteria. Examples of this type of competition are the FIFA World Cup group stage and the NBA regular season.
- Elimination tournaments: Elimination tournaments are a type of tournament where participants are paired up and play against each other in a single game or a series of games. The winner advances to the next round, while the loser is eliminated. The process is repeated until there is only one champion. Elimination tournaments are also known as knockout tournaments. Examples of this type of competition are the FIFA World Cup knockout stage and NBA playoffs.
- Hybrid tournaments: A combination of round-robin and elimination formats, where participants first play in groups or pools and then advance to a knockout phase based on their performance. Examples of this type of competition are the UEFA Champions League and the Olympic Games.

Many researchers tend to create scheduling algorithms tailored to the specific needs of individual sports leagues. However, this has led to a shortage of studies that compare how well these algorithms perform in practice. To address this gap, Van Bulck et al. introduced a standardized data format for round-robin sports timetabling in the RobinX project [2]. This format makes it easier to evaluate and compare different scheduling algorithms, providing valuable insights for sports organizers and stakeholders.

2.3 Solution Methods

Combinatorial Optimization Problems (COPs) are diverse and can vary greatly in terms of their nature and complexity. Consequently, a range of methods and techniques have been developed to tackle these challenges effectively. These methods are chosen based on the specific problem at hand and the desired trade-off between solution quality and computational efficiency [9, 16].

2.3.1 Exact algorithms

Exact algorithms, also known as deterministic algorithms, are a category of optimization techniques designed to discover the optimal solution within a finite amount of time. In sports scheduling, exact methods are based on mathematical models and algorithms and guarantee to find the optimal solution or prove its nonexistence. Exact methods include integer programming, constraint programming, and branch-and-bound algorithms. These methods are usually very efficient for small and medium-sized instances of simple COPs, but they may become impractical for large and complex problems due to the exponential growth of the search space.

2.3.2 Heuristic algorithms

Heuristic algorithms serve as problem-solving techniques for addressing combinatorial optimization problems. These methods prioritize finding a good solution within a reasonable time frame rather than guaranteeing an optimal solution. They become particularly valuable when dealing with exceedingly complex or large problems where obtaining exact solutions is either impractical or unnecessary. Diverse categories of heuristic algorithms exist, differentiated by their approaches to navigating the search space and assessing solution quality. Some of the most common types are:

Constructive heuristics

These algorithms generate a solution step by step, starting from scratch and gradually building a complete solution. The greedy algorithm is an example, which consistently selects the best available element at each stage based on a specific criterion. Constructive heuristics often exhibit speed but may occasionally become trapped in local optima.

Improvement heuristics

These algorithms start with an initial solution and try to make it better by making small changes like swapping, adding, or removing elements. In combinatorial optimization, a *neighborhood* refers to a set of solutions that are closely related to a given solution. These related solutions are typically obtained by making a small, well-defined change to the original solution. For example, the local search algorithm is an improvement heuristic that keeps moving towards a better nearby solution until it cannot find any more improvements. While improvement heuristics can improve solution quality, they can also get stuck in local optima.

2.3.3 Metaheuristics

These algorithms combine or adapt other heuristics to escape local optima and more efficiently explore the search space. An example is the simulated annealing algorithm, which emulates the physical annealing process by permitting occasional sub-optimal moves with a decreasing probability over time. While metaheuristics can often identify nearly optimal solutions for various problems, they might demand greater computational resources and parameter fine-tuning than heuristic methods.

2.3.4 Matheuristic

Matheuristics[7] represent optimization algorithms that combine mathematical programming techniques with heuristic methods, including metaheuristics, to solve complex and large-scale combinatorial problems. These methods aim to blend the precision of mathematical programming with the speed and adaptability of heuristics. One type of matheuriscs is Improvment Matheuristics. These matheuristics initiate from an initial feasible solution and employ mathematical programming to enhance it by adjusting specific elements of the solution. For instance, an improvement heuristic for the traveling salesman problem might utilize a mixed-integer program to optimize a subset of nodes within the existing tour while keeping the remainder fixed. Improvement heuristics can elevate solution quality but are often reliant on the quality of the initial solution.

2.4 Terminalogy

To enhance the clarity of the thesis, we provide fundamental terminology used in sports timetabling. The RobinX paper[2] offers a comprehensive overview of the common terminology employed in sports scheduling problems. In the following sections, we will delve into the specific concepts essential for comprehending the characteristics of the problem instances used in this thesis.

In a *round-robin tournament*, each team competes against every other team a set number of times. Typically, many sports leagues use a double round-robin (2RR) format, where teams face each other twice. However, it is worth noting that single, triple, and even quadruple round-robin tournaments are also found in some instances available in the literature.

When organizing a tournament, it is essential to assign the games to a certain number of *time slots* (referred to as *slots*). The aim is to ensure that each team participates in no more than one game during each slot. The number of slots needed for scheduling a single round-robin tournament depends on the total number of teams, denoted as n.

1. When the number of teams is even, a minimum of $(n-1)$ slots are required to schedule the tournament effectively.
2. When the number of teams is odd, at least n slots are required to accommodate a single round-robin tournament.

Additionally, we distinguish between two scheduling scenarios:

The tournament is *compact* when the number of available slots matches the lower bound required to schedule the tournament. In other words, all the games are planned to take place in exactly the minimum number of slots required for the given number of teams. The tournament is called *relaxed* when there are more slots available than the minimum required for scheduling the tournament. In this case, having extra slots beyond what is strictly necessary for the tournament's completion is possible.

In sports scheduling literature, teams are usually associated with specific *venues*. When a team plays at its designated venue, those games are called *home games*. On the contrary, when they play at any other venue, those games are considered *away games*. In time slots where there are no scheduled games for a team, it is referred to as a *bye*. It is assumed that whenever two teams face each other, one of them plays at their home venue, and the other plays away.

In the case of single round-robin schedules, it is often a requirement that the difference between the number of home games and away games played by each team is no greater than 1. When this condition is met, the schedule is referred to as *balanced*. For double round-robin schedules, it is typically mandated that the two games between any pair of opponents take place at opposite venues. This requirement automatically ensures a balanced schedule because each team plays an equal number of home and away games against the same opponent.

A team is said to have a *break* when they play two games consecutively either at home or away. In other words, a break happens when a team plays two back-to-back games with the same home-away status. In this competition, we note the occurrence of a break in the time slot of the second consecutive game. For example, if Team 1 plays a home game in slot 1 and another home game in slot 2, a break is recorded in slot 2 because they have played two consecutive home games.

In a *phased* tournament, the schedule is divided into two halves. Each half forms its single round-robin tournament, ensuring that every team plays against every other team once within that half. This process is repeated separately for the first and second halves of the schedule. In contrast, an *unphased* tournament does not require matches to follow specific constraints where there are no strict requirements for forming complete round-robin tournaments within each half of the slots, offering more scheduling flexibility.

2.5 Related works

The instances that were used in this thesis were part of the International Timetabling Competition on Sports Timetabling [19]. Participants in this competition have used various approaches to solve each instance. One of the frequently employed strategies, commonly known as the fix-and-optimize approach, falls under the category of matheuristics. This strategy involves iteratively utilizing a mathematical programming solver to optimize a specific subset of variables while the remainder of the variables remain fixed [20].

The approach introduced by Phillips et al.[8], employs an Adaptive Large Neighborhood Search (ALNS) matheuristic. This approach can be similar to a *fix-and-optimize* method, enhanced by an adaptive control strategy inspired by reinforcement learning. In the initial phase, the algorithm attempts to solve a comprehensive integer programming problem. If it fails to do so within a predetermined time frame, it switches to a canonical factorization method proposed by de Werra[3]. Subsequently, the algorithm dedicates the remaining time to the ALNS technique, which involves modifying only a portion of the solution in each iteration. Within each sub-problem of the neighborhood, an integer program is solved to minimize the number of violations of both hard and soft constraints. What makes this algorithm stand out is its adaptive approach to defining the neighborhood. This adaptability is treated as a multi-armed bandit problem, utilizing the upper confidence bound method [15]. In essence, it dynamically selects from a range of different neighborhood types and sizes based on past performance. It keeps a balance between exploring different neighborhoods and exploiting those that have demonstrated effectiveness. Notably, the study found that searching through a greater number of smaller neighborhoods proved to be the most efficient strategy.

Fonseca and Toffolo [4] employed a comparable approach using the fix-and-optimize method. In this algorithm, which utilizes an initial solution generated through the Polygan Method [10], the process unfolds in two distinct phases. Initially, the algorithm runs to secure a feasible solution, focusing exclusively on hard constraints. Once a feasible solution is obtained in the first phase, the algorithm proceeds to its second phase. During this second phase, all slack variables related to hard constraints are eliminated, and the soft constraints are introduced into the model, facilitating further refinement and improvement of the solution.

The metaheuristic proposed by Rosati et al.[11] employs six local search neighborhoods, including five previously established ones (SwapHomes, SwapTeams, SwapRounds, PartialSwapTeams, and PartialSwapRounds) and an innovative one called PartialSwapTeamsPhased, designed to perform partial swaps among opponents of two teams while maintaining phase constraints. These neighborhoods are integrated using simulated annealing, incorporating a cut-off mechanism for faster early-phase search. The algorithm conducts three sequential simulated annealing runs, aiming to find feasible solutions without violating the hard constraints, explore both feasible and infeasible regions, and finally, locate improved local minima within the feasible region. The first stage starts from a greedily-constructed solution, and subsequent stages build upon the best solution from the previous one. Importantly, the algorithm operates based on the number of local search evaluations rather than fixed time limits, resulting in varying running times depending on instance size and constraint complexity.

The FBHS (first-break-heuristically-schedule) algorithm, as proposed by Van Bulck and Goossens [18], takes a different approach to schedule by sequentially solving two key sub-problems: determining the home and away schedules for each team in every time slot, i.e. Home and Away Pattern (HAP), and subsequently, establishing the opponents for each team in each time slot (opponent schedule). Ensuring compatibility between these two steps is crucial, as teams can only compete against each other when their home and away schedules align. To create the HAP set, the algorithm employs Benders' decomposition, enforcing necessary feasibility conditions while considering the LP relaxation of the optimal opponent schedule. Once a promising HAP set is obtained, a compatible opponent schedule is constructed using a fix-and-optimize matheuristic.

Chapter 3

Problem Description

Let us denote the set of participating teams in the 2RR as T and the set of time slots (rounds) as S. In the instances used in this thesis, n (the number of teams) is even and the set of time slots is compact, hence we can determine that the cardinality of S, denoted as $|S|$, is equal to $2n-2$. In all problem instances considered, n takes values of either 16, 18, or 20. This choice aligns with real-life scenarios and represents problem sizes that typically challenge state-of-the-art optimization techniques.

In addition to the structural constraints mandating the scheduling of all games within the 2RR and ensuring that no team plays more than one game per time slot, the problem instances incorporate nine constraint types in which Van Bulck et al. [2] believe that this selection of constraint types covers the majority of real-life scheduling constraints. Constraints can be categorized as either *hard* or *soft*, where hard constraints represent non-negotiable properties of the timetable that must always be upheld, and soft constraints express preferences that should be satisfied whenever possible.

The primary objective in the problem instances is to minimize the total (weighted) sum of deviations resulting from violated soft constraints while respecting all hard constraints.

3.1 Constraints

There are a total of nine constraint types, which can be grouped into the following five constraint classes:

3.1.1 Capacity constraints

Capacity constraints serve to determine whether a team plays at home or away and control the total number of games a team or a group of teams can play during a specified time period. There are four distinct types of capacity constraints (CA1, CA2, CA3, CA4), each of which can be either considered as hard or soft constraints.

- **CA1 Constraints:** These constraints set an upper limit on the number of home games or away games a particular team can play during a given set of time slots. They are used to model situations where teams cannot play at home during specific time slots, such as when stadiums are unavailable. CA1 constraints also help in achieving a balance between home and away games for teams throughout the season.
- **CA2 Constraints:** CA2 constraints are a extension of CA1 constraints. They establish an upper limit on the number of home games or away games for a given team against a specific set of other teams during a specified time slot. For example, they can limit the number of away games a lower-ranked team plays against stronger teams during the later part of the season.
- **CA3 Constraints:** CA3 constraints place restrictions on the maximum sequence of consecutive home or away games against specific teams. In selected instances, a hard CA3 constraint ensures that no team plays more than two consecutive home games or away games. When they are considered as soft constraints, they indicate that a team should not have more than two home games, away games, or games (home or away) against a specific set of teams, typically based on their strength group, within every four consecutive rounds.
- **CA4 Constraints:** CA4 constraints set an upper limit on the number of home games, away games, or games (home or away) between teams from one set against teams from another set during a specified time slot. These constraints can be used to restrict the number of games between teams from the same strength group or limit the total number of home games for a set of teams during a particular time slot, perhaps because teams share a stadium or are geographically close.

3.1.2 Game constraints

A GA1 constraint, whether categorized as hard or soft, has the purpose of either mandating or prohibiting the scheduling of a particular game or a set of games during specific time slots. These constraints are versatile and can be employed to address various scheduling scenarios. For instance, they can enforce rules such as avoiding scheduling high-risk games during time slots when other major events are planned. Additionally, they can dictate that certain games must take place during designated *derby time slots*. It is important to note that within the RobinX framework, the GA1 constraint is the sole type of game constraint that is taken into consideration for the scheduling instances.

3.1.3 Break constraints

Constraints BR1 can be designated as either hard or soft constraints, and their purpose is to set an upper limit on the total number of breaks that a specific team can have during a designated set of time slots. These constraints are versatile and can be applied to ensure, for example, that there are no breaks near the beginning or end of the season.

A BR2 constraint is used to restrict the overall number of breaks within the timetable. Therefore, there is a maximum of one BR2 constraint per instance, which can be either hard or soft.

When CA3 constraints are expressed as hard constraints, they implicitly prevent any team from having two consecutive breaks. This contributes to the overall structure and fairness of the scheduling.

3.1.4 Fairness constraints

This set of constraints is designed to promote fairness in the tournament schedule. It emphasizes the concept of *2-ranking balance*, which implies that at any given point in time, no two teams should have a difference of more than two home games played. In other words, it ensures that the distribution of home games among teams remains relatively equitable throughout the schedule. This constraint is valuable in preventing situations where certain teams consistently have more or fewer home games, which could affect the fairness of the competition.

3.1.5 Separation constraints

SE1 constraints address the need for temporal spacing between games involving the same teams. They promote the idea that games between the same teams should be sufficiently spaced apart. In the context of the RobinX problem instances, these constraints suggest that there should be a minimum of 10 time slots between successive games involving identical teams. This spacing ensures that teams have adequate time to prepare for their next encounter and adds an element of anticipation and excitement for fans, making the tournament more engaging.

3.2 Instances

There are 45 instances that we attempted to solve in this study. The number of teams can be either 16, 18, or 20. The number of constraints varies from 93 in instance 3_15 to 1486 in instance 2_2, but this number does not necessarily reflect the difficulty of the instance. Table 3.1 offers a comprehensive overview, detailing the number of teams, phased status, constraint types, and the number of constraints available for each of the instances utilized in this study. These instances were generated in three phases, which is reflected in their naming conventions.

Table 3.1: Instances overview

Inst.	#Teams	Phased	Constraint Types	#Constr.
1_1	16	Yes	BR1, BR2, CA1, CA2, CA4, FA2, GA1, SE1	206
1_2	16	Yes	BR1, BR2, CA1, CA3, FA2, GA1	168
1_3	16	Yes	BR1, BR2, CA1, CA2, CA3, FA2, GA1	335
1_4	18	Yes	BR1, BR2, CA1, CA2, CA4, GA1, SE1	441
1_5	18	Yes	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	803
1_6	18	Yes	BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	999
1_7	18	No	BR1, BR2, CA1, CA2, CA4, GA1, SE1	1343
1_8	18	No	BR1, CA1, CA2, CA3, CA4, FA2, GA1	653
1_9	18	No	BR1, BR2, CA1, CA2, CA3, FA2, GA1	193
1_10	20	Yes	BR1, BR2, CA1, CA2, CA3, CA4, SE1	1270
1_11	20	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1363
1_12	20	Yes	BR1, BR2, CA1, CA2, CA3, CA4, GA1	214

1_13	20	No	BR1, BR2, CA1, CA2, CA3, GA1	532
1_14	20	No	BR1, BR2, CA1, FA2, GA1	113
1_15	20	No	BR1, BR2, CA1, CA2, CA3, CA4, FA2, GA1	1412
2_1	16	Yes	BR1, BR2, CA1, CA2, CA4, SE1	1146
2_2	16	Yes	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1486
2_3	16	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	1459
2_4	18	Yes	BR1, CA1, CA2, CA3, CA4, GA1	265
2_5	18	Yes	BR1, BR2, CA1, CA2, CA3, FA2, GA1	349
2_6	18	Yes	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	325
2_7	18	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	626
2_8	18	No	BR1, CA1, CA2, CA3, CA4, GA1	286
2_9	18	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1, FA2, GA1	296
2_10	20	Yes	BR1, BR2, CA1, CA2, CA4, GA1	912
2_11	20	Yes	BR1, CA1, CA2, CA4, GA1	1225
2_12	20	Yes	BR1, BR2, CA1, CA2, CA3, FA2, GA1, SE1	314
2_13	20	No	BR1, CA1, CA2, CA3, FA2, GA1, SE1	578
2_14	20	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	881
2_15	20	No	BR1, BR2, CA1, CA2, CA3, FA2, GA1, SE1	237
3_1	16	No	BR1, CA1, CA2, CA3, CA4, FA2 GA1	778
3_2	16	No	BR1, BR2, CA1, CA2, CA3, CA4, GA1	1323
3_3	16	No	BR1, BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	576
3_4	18	Yes	BR1, CA1, CA4, GA1, SE1	139
3_5	18	Yes	BR2, CA1, CA2, CA3, CA4, FA2, GA1	924
3_6	18	Yes	BR1, BR2, CA1, CA2, CA4, FA2, GA1, SE1	331
3_7	18	No	BR1, BR2, CA1, CA2, CA3, GA1, SE1	873
3_8	18	Yes	BR1, BR2, CA1, CA2, CA3, GA1, SE1	314
3_9	18	No	BR1, BR2, CA1, CA2, CA3, FA2, GA1	505
3_10	20	Yes	BR1, BR2, CA1, CA2, CA3, CA4, GA1, SE1	936
3_11	20	Yes	BR1, BR2, CA1, CA2, CA3, FA2, GA1	419
3_12	20	No	BR1, BR2, CA1, CA2, CA3, CA4, SE1	1262
3_13	20	No	BR2, CA1, CA2, CA3, CA4, FA2, GA1, SE1	313
3_14	20	No	BR1, CA1, CA2, CA3, CA4, FA2, GA1	1110
3_15	20	No	BR1, BR2, CA1, CA3, FA2, GA1	93

Chapter 4

Model Formulation

4.1 Mathematical model

Here we give a mathematical form of the problem. Table 4.1 serves as a comprehensive reference for all the notation utilized within the mathematical model. Constraints can be tailored to apply exclusively to home games (H), away games (A), or both (HA), denoted by the use of superscripts. For instance, if a capacity constraint of type one is designed specifically for home games, it is represented in the model as $CA1^H$. This notation reflects the constraint's targeted scope. Constraints are categorized into six categories and described below.

4.1.1 Structural Constraints

The foundational elements of our model are the structural constraints, which form the backbone of the entire framework. Among the set of constraints considered, constraints 4.1 through 4.8 are consistently present in all 45 instances investigated in this study. However, it is important to note that constraints 4.9 and 4.10 are exclusively applicable to the phased instances, setting them apart from the rest in terms of their constraint configuration.

Constraint 4.1 is incorporated for the purpose of simplifying notation and facilitating code implementation and it prevents any game between a team and itself from happening. Constraint 4.2 guarantees that each team participates in precisely one game during each

Sets	
T	Set of teams
T_c^1	First indexed subset of teams for every constraint
T_c^2	Second indexed subset of teams for every constraint
S	Set of all slots
S_c	Indexed Subset rounds for every constraint
G_c	Indexed multiset of ordered pairs (i, j) for every constraint
C	Set of all Constraints
Parameters	
t_c	The maximum value that some linear combination of the variables can take without incurring a penalty for the non-structural constraint c.
d_c	an integer deviation variable for non-structural constraint c. $d_c = 0$ if a constraint c is hard. These variables are referred to by slack variables.
Variables	
x_{ijs}	Binary; 1 if the match (i, j) is played in slot s; 0 otherwise. $\forall i, j \in T, \forall s \in S$
h_{is}	Binary; 1 if i has a home break in slot s; 0 otherwise. $\forall i \in T, \forall s \in S \setminus \{1\}$
a_{is}	Binary; 1 if i has an away break in slot s; 0 otherwise. $\forall i \in T, \forall s \in S \setminus \{1\}$
y_{ij}	Binary; 1 if the match (i, j) occurs before match (j, i);0 otherwise. $\forall i, j \in T$

Table 4.1: notation used in model

time slot. Constraint 4.3 ensures that every ordered pair of teams encounters each other exactly once throughout the entire season. Essentially, this constraint guarantees the allocation of all games in a season to specific time slots.

$$\sum_{s \in S} x_{iis} = 0 \qquad \forall i \in T \quad (4.1)$$

$$\sum_{j \in T \setminus \{i\}} (x_{ijs} + x_{jis}) = 1 \qquad \forall i \in T, \forall s \in S \quad (4.2)$$

$$\sum_{s \in S} x_{ijs} = 1 \qquad \forall i, j \in T, i \neq j \quad (4.3)$$

$$x_{ijs}, h_{is}, a_{is}, y_{ij} \in \{0, 1\} \qquad \forall i, j \in T, s \in S \quad (4.4)$$

As certain constraints rely on the break status and the number of breaks for teams, represented by the variables h_{is} and a_{is}, Constraints 4.5 and 4.6 have been introduced to establish a connection between these variables and the x_{ijs} variables. This linkage ensures that h_{is} and a_{is} attain the correct values as required by the model.

$$\sum_{j \in T}(x_{ijs} + x_{i,j,s-1}) \leq h_{is} + 1 \qquad \forall i \in T, \forall s \in S \setminus \{1\} \quad (4.5)$$

$$\sum_{j \in T}(x_{jis} + x_{j,i,s-1}) \leq a_{is} + 1 \qquad \forall i \in T, \forall s \in S \setminus \{1\} \quad (4.6)$$

The implementation of Separation constraints in section 4.1.6 becomes more straightforward when we can ascertain that the match (i, j) took place before or after the match (j, i). To this end, the variables y_{ij} are introduced into the model. Constraints 4.7 and 4.8 establish the relationship between y_{ij} and the x_{ijs} variables.

$$\sum_{s \in S} s(x_{jis} - x_{ijs}) \leq M y_{ij} \qquad \forall i, j \in T \quad (4.7)$$

$$\sum_{s \in S} s(x_{ijs} - x_{jis}) \leq M(1 - y_{ij}) \qquad \forall i, j \in T \quad (4.8)$$

where M is a constant that satisfies the condition $M \geq |S| - 1$.

A subset of the instances is characterized as phased, indicating that the league's overall structure is composed of two consecutive 1RRs (round robins). Constraints 4.9 and 4.10 are employed to ensure that each pair of teams engages in one match during the first half of the season and another during the second half of the season.

$$\sum_{s=1}^{|S|/2}(x_{ijs} + x_{jis}) = 1 \qquad \forall i, j \in T, i \neq j \quad (4.9)$$

$$\sum_{k=|S|/2+1}^{|S|}(x_{ijs} + x_{jis}) = 1 \qquad \forall i, j \in T, i \neq j \quad (4.10)$$

4.1.2 Capacity Constraints

The set of CA1 constraints places restrictions on the maximum number of games in which a specified subset of teams can participate within a designated subset of slots. For instance, a team from T_c^1 is limited to playing a maximum of t_c games within the specified subset of slots known as S_c. Constraints 4.11 and 4.12 govern the count of home (H) or away (A) games for each team i in T_c^1 during the subset of slots defined in S_c. These constraints apply to all hard constraints, and d_c keeps track of the number of times each soft constraint is violated.

$$\sum_{j \in T \setminus i} \sum_{s \in S_c} x_{ijs} \leq t_c + d_{ic} \qquad \forall c \in CA1^H, i \in T_c^1 \quad (4.11)$$

$$\sum_{j \in T \setminus i} \sum_{s \in S_c} x_{jis} \leq t_c + d_{ic} \qquad \forall c \in CA1^A, i \in T_c^1 \quad (4.12)$$

The set of CA2 constraints can be viewed as an extension of CA1, with the additional specification of the opponent team. For instance, Team i from T_c^1 is constrained to play a maximum of t_c games within the designated subset of slots S_c, against teams j in T_c^2. CA2 constraints also introduce an additional mode, HA, which accounts for the total number of games played, encompassing both home and away matches.

$$\sum_{j \in T_c^2} \sum_{s \in S_c} (x_{ijs} + x_{jis}) \leq t_c + d_{ic} \qquad \forall c \in CA2^{HA}, i \in T_c^1 \quad (4.13)$$

$$\sum_{j \in T \setminus i} \sum_{s \in S_c} x_{ijs} \leq t_c + d_{ic} \qquad \forall c \in CA2^H, i \in T_c^1 \quad (4.14)$$

$$\sum_{j \in T \setminus i} \sum_{s \in S_c} x_{jis} \leq t_c + d_{ic} \qquad \forall c \in CA2^A, i \in T_c^1 \quad (4.15)$$

CA3 constraints (4.16, 4.17, 4.18) specify that each team i in T_c^1 is limited to playing a maximum of t_c home games, away games, or games (home or away) against teams j in T_c^2 within each sequence of $intp_c$ consecutive time slots. For example, Team 0 (from T_c^1) can engage in at most two consecutive matches against Team 1, 2, and 3 (from T_c^2) within each sequence of 3 ($intp_c$) time slots. To illustrate, if a match x_{011} (where Team 0 plays at home against Team 1 in slot 1) occurs, then Team 0 is restricted to playing only one more game against Team 2 or 3 in the subsequent two slots (forming a sequence of 3).

$$\sum_{j \in T_c^2} \sum_{l=k}^{k+intp_c-1} (x_{ijs} + x_{jis}) \leq t_c + d_{ikc} \qquad \forall c \in CA3^{HA}, i \in T_c^1, k \in S : k \leq |S| - intp_c + 1 \quad (4.16)$$

$$\sum_{j \in T \setminus i} \sum_{l=k}^{k+intp_c-1} x_{ijs} \leq t_c + d_{ikc} \qquad \forall c \in CA3^H, i \in T_c^1, k \in S : k \leq |S| - intp_c + 1 \quad (4.17)$$

$$\sum_{j \in T \setminus i} \sum_{l=k}^{k+intp_c-1} x_{jis} \leq t_c + d_{ikc} \qquad \forall c \in CA3^A, i \in T_c^1, k \in S : k \leq |S| - intp_c + 1 \quad (4.18)$$

The CA4 constraints encompass two distinct modes, known as *global* and *every*. These modes are applied to specific subsets of teams, denoted as i in T_c^1 and k in T_c^2. The *global* mode, represented by constraints 4.19 through 4.21, is utilized to restrict the overall number of games played in the tournament, while *every* mode, encompassed by constraints 4.22 through 4.24, imposes limitations on the total number of games played during each slot within a specified subset S_c. To illustrate, if we consider a subset of rounds as $\{1, 2, 3\}$, *global* mode restricts the total number of games played throughout the entire tournament, whereas *every* mode limits the number of games played in each of the slots 1, 2, and 3 individually.

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{s \in S_c} (x_{ijs} + x_{jis}) \leq t_c + d_c \qquad \forall c \in CA4g^{HA} \quad (4.19)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{s \in S_c} x_{ijs} \leq t_c + d_c \qquad \forall c \in CA4g^{H} \quad (4.20)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} \sum_{s \in S_c} x_{jis} \leq t_c + d_c \qquad \forall c \in CA4g^{A} \quad (4.21)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} (x_{ijs} + x_{jis}) \leq t_c + d_{sc} \qquad \forall c \in CA4e^{HA}, \forall s \in S_c \quad (4.22)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} x_{ijs} \leq t_c + d_{sc} \qquad \forall c \in CA4e^{H}, \forall s \in S_c \quad (4.23)$$

$$\sum_{i \in T_c^1} \sum_{j \in T_c^2} x_{jis} \leq t_c + d_{sc} \qquad \forall c \in CA4e^{A}, \forall s \in S_c \quad (4.24)$$

4.1.3 Game Constraints

The set of game constraints is denoted as GA. GA1 specifies that during time slots s in S_c, there must be at least t'_c and at most t_c games from the multiset G_c. The multiset G_c

comprises ordered pairs (i, j), where i represents the home team hosting the game, and j represents the away team.

$$\sum_{(i,j)\in G_c} \sum_{s\in S_c} x_{ijs} \leq t_c + d_c \qquad \forall c \in GA \quad (4.25)$$

$$\sum_{(i,j)\in G_c} \sum_{s\in S_c} x_{ijs} \geq t'_c - d_c \qquad \forall c \in GA \quad (4.26)$$

For each constraint c within GA, both lower and upper bound constraints are established. Notably, if one of these constraints has a positive d_c value, it implies that the other constraint is satisfied as a strict inequality.

4.1.4 Break Constraints

BR1 serves the purpose of preventing breaks at the start or end of the season while also constraining the total number of breaks for each team. Constraints 4.27-4.29 monitor deviations, which arise when a team in T_c^1 experiences more than $intp_c$ breaks (HA, H, or A) during the specified round(s) in S_c. If c represents a soft constraint, these deviations are counted; otherwise, they enforce that teams have no more breaks than the specified limit.

$$\sum_{s\in S_c} (h_{is} + a_{is}) \leq intp_c + d_{ic} \qquad \forall c \in BR1^{HA}, i \in T_c^1 \quad (4.27)$$

$$\sum_{s\in S_c} h_{is} \leq intp_c + d_{ic} \qquad \forall c \in BR1^{H}, i \in T_c^1 \quad (4.28)$$

$$\sum_{s\in S_c} a_{is} \leq intp_c + d_{ic} \qquad \forall c \in BR1^{A}, i \in T_c^1 \quad (4.29)$$

BR2 calculates the total number of breaks (considering HA mode exclusively) for all teams in T, ensuring that this count remains less than or equal to $intp_c$ for time slots $s \in S$. Constraint 4.30 imposes limitations on the overall number of breaks in the season, or it tracks instances where the total number of breaks surpasses $intp_c$.

$$\sum_{i\in T_c^1} \sum_{s\in S} (h_{is} + a_{is}) \leq intp_c + d_c \qquad \forall c \in BR2^{HA} \quad (4.30)$$

4.1.5 Fairness Constraints

FA2 specifies that the difference in the number of home games (only mode available in selected instances) played by any two teams up to a given slot should not surpass a predefined maximum. This constraint aims to maintain a balance in the distribution of home games across slots for all pairs of teams.

Constraints 4.31 and 4.32 serve the purpose of monitoring the count of deviations, provided that c represents a soft constraint, and they ensure that the difference does not exceed the specified maximum limit in the case of c being a hard constraint.

$$\sum_{l=0}^{s} \sum_{h \in T_c^1} (x_{ihs} - x_{jhs}) \leq intp_c + d_{ijc} \qquad \forall c \in FA2, s \in S_c, i, j \in T_c^1 : i < j \quad (4.31)$$

$$\sum_{l=0}^{s} \sum_{h \in T_c^1} (x_{jhs} - x_{ihs}) \leq intp_c + d_{ijc} \qquad \forall c \in FA2, s \in S_c, i, j \in T_c^1 : i < j \quad (4.32)$$

4.1.6 Separation Constraints

A collection of separation constraints, denoted by SE, is designed to prevent matches between a pair of teams from occurring too closely in time. Each constraint c within SE defines a set T_c^1 from which pairs of teams are chosen. Constraint 4.33 specifies that for each pair (i, j) of teams in T_c^1, there should be a minimum of t_c time slots between two matches involving the same opponents.

$$\sum_{s \in S} s(x_{ijs} - x_{jis}) \geq t_c + 1 - d_c - My_{ij} \qquad \forall c \in SE, \forall i, j \in T_c, i \neq j \quad (4.33)$$

where M is a constant that satisfies the condition $M \geq |S| - 1$.

4.1.7 Objective Function

Let $C = CA1 \cup CA2 \cup CA3 \cup CA4 \cup GA1 \cup BR1 \cup BR2 \cup FA2 \cup SE$, which represents the set of all constraints in the problem, excluding the structural constraints outlined in Section 4.1.1 . We can further divide C into two subsets: $\tilde{C}$, comprising the soft constraints, and $\bar{C}$, consisting of the hard constraints. Each constraint $c \in \tilde{C}$ has a given unit penalty denoted as w_c. Consequently, the objective function can be expressed as follows:

Minimize $\sum_{c \in \tilde{C}} w_c d_c$

It is important to note that we set $d_c = 0$ for all $c \in \bar{C}$ to ensure feasibility.

Chapter 5

Solution Approach

5.1 Implementation

The data provided by the RobinX project is already structured in a human-readable XML format. However, it is essential to reorganize this data into a format that aligns with the requirements of the solver. To achieve this, we have developed a method that parses the XML file for each instance, categorizing and storing each constraint type separately. Given that each constraint targets different variables, it is crucial to distinguish constraints based on their modes (H, A, HA, $every$, $global$). Once this categorization is complete, the model is generated. It begins with incorporating structural constraints and is followed by including the previously categorized constraints. This process results in the creation of a ready-to-use Integer Linear Programming (ILP) model for the solver.

Recognizing that solving the primary model could potentially entail weeks of computation without guaranteeing an optimal solution, this thesis adopts a two-step approach. Initially, the focus lies on finding an initial solution for each instance. Subsequently, the research leverages a range of heuristic methods in the second step to enhance these initial solutions.

5.2 Obtaining an initial solution

In addressing the diverse complexities of the instances, we have pursued various approaches to attain feasible initial solutions. The first attempt involved tackling the entire

Integer Linear Programming (ILP) model without permitting any hard constraint violations. To prioritize the attainment of any feasible solution over improving it, we have configured the solver accordingly. Due to the extensive solution space, none of the instances yielded a feasible solution within the twenty-four-hour run-time allocated.

The second approach centered on solving the primary model while disregarding the soft constraints, effectively setting the objective function to zero. Here, the primary aim was to secure a feasible solution, and the search would halt upon achieving this goal. In the third attempt, we treated the hard constraints as soft constraints, excluding the soft constraints from the objective function. Consequently, the objective function in this scenario became a weighted sum of hard constraints, with the objective of reaching a value of zero. The final approach resembled the previous attempt, but it incorporated the adaptive heuristic to explore sub-problems. Further details on this adaptive heuristic and its role in improving initial solutions are provided in the subsequent section.

5.3 Improving Solutions

In this phase, the primary approach involved conducting neighborhood searches, allowing for the modification of a portion of the solution while keeping the remainder unchanged.

5.3.1 Heuristics

Two broad types of heuristics, or neighborhoods, were defined. In the first type, a selection of time slots (slots) was chosen, and all variables associated with those slots became the decision variables in the sub-problem. Variables linked to other slots were held constant at their current values within the solution. In the second type of neighborhood, a subset of teams was selected, and all variables related to those teams were considered as decision variables, while the rest remained fixed.

5.3.2 Neighborhood Selection

Three distinct selection methods were employed. The first method, emphasizing diversification, involved random selection of neighborhoods. This approach facilitated exploration in various regions of the solution space.

The second method assessed the impact of each decision variable (those with a value of 1) on slack variables used to measure the violation of soft constraints. Each decision variable was associated with two teams and one time slot. The frequency with which each team and time slot appeared in this analysis was used as a measure of their contribution to the total penalty of the solution, thus determining their weight in the selection of decision variables. This method prioritized teams/slots with significant contributions to the overall penalty, focusing on intensification. In the rest of this thesis, we refer to this method as *costliest_ones*.

The third method resembled the second one, but it considered all decision variables on the left-hand side of violated soft constraints, not just those with a value of 1. This approach offered greater flexibility than the second method and less flexibility than random selection, aiming to balance both intensification and diversification. If the number of visible teams/slots exceeded the requirement, a random subset was selected. Conversely, in cases where additional teams or slots were needed, they were chosen randomly. In the rest of this thesis, we refer to this method as *costliest_all*.

5.3.3 Neighborhood Size

This thesis explored two distinct ways to determine neighborhood sizes: fixed and dynamic. In the fixed-size approach, a predetermined number of teams/slots were designated as decision variables, while the remaining variables were held constant at their existing values. This fixed size remained unchanged until the time limit was reached. Specifically, the size constituted 1/3 of the total number of teams for team neighborhoods and 1/4 of the total number of slots for slot neighborhoods.

The dynamic size approach commenced with the same size as the fixed size. However, it introduced the possibility of adjusting the size based on the solver's performance in the sub-problems. In the final chosen configuration, it was established that if ten consecutive subproblems were solved in less than 60 seconds, the neighborhood size would increase by one. Conversely, if one sub-problem reached the time limit, the neighborhood size would decrease by one. It is important to note that changes in neighborhood size were tracked independently for team and slot neighborhoods.

5.3.4 Heuristic combination

In this phase of the study, various combinations of the previously defined heuristics were examined to determine their effectiveness in improving solutions. The goal was to find the most efficient combination of heuristics to enhance the overall performance of the algorithm. To ensure a fair comparison, all of these combinations shared the same total time limit, and the final result represented the output of all iterations possible within that time frame. Furthermore, within each iteration, the time limit allocated to individual heuristics depended on whether they operated with smaller or larger sub-problems. Specifically, if a heuristic involved smaller sub-problems, it was granted a time limit of 300 seconds, whereas heuristics dealing with larger sub-problems (specifically those defined in Double Heuristics) were allotted 600 seconds per iteration. The total time limit for attempting to find an optimal solution for a given instance was capped at 4800 seconds.

During the improvement process, the initial solution obtained in the previous phase served as the starting point for the first iteration. Subsequent iterations employed a warm start strategy, where the output of the previous iteration became the initial solution for the next.

Single heuristic

In this approach, the focus was on systematically evaluating the performance of individual heuristics on each instance. A total of 12 unique combinations were tested, each involving the application of a single heuristic. These combinations encompassed a range of strategies, including fixing either teams or slots, selecting neighborhoods randomly, prioritizing the selection of the variables in the costliest_all, or costliest_ones. Additionally, the size of the neighborhood could be either fixed or dynamic, as explained in sections 5.3.2 and 5.3.3.

Double heuristics

In this approach, the study explored the impact of using both slot and team heuristics consecutively within the same iteration. This meant that, in each iteration, one slot heuristic and one team heuristic were applied sequentially to refine the solution. Notably, both the slot and team heuristics shared the same criteria for neighborhood selection and size. For instance, one combination involved applying a slot heuristic followed by a team heuristic, with both heuristics selecting one-fourth of the slots or teams for exploration while keeping the rest fixed. This selection process was randomized for both heuristics.

Furthermore, this phase introduced two new heuristics into the mix. Unlike the previous small sub-problem approach, these new heuristics operated with larger sub-problems. They achieved this by fixing only one-fourth of the teams or slots, allowing the remaining three-fourths to be explored and optimized. Overall, four distinct combinations were tested in this phase: random selection of neighborhoods, prioritizing the selection of the variables in the costliest_all, or costliest_ones, and the configuration mentioned earlier, but with random selection for both slot and team heuristics.

Adaptive heuristic

In this approach, a more dynamic and adaptive strategy was implemented. All the heuristics introduced in sections 5.3.1-5.3.3 were made available for selection in each iteration, with the choice of heuristic being made randomly based on specific probabilities. Initially, all heuristics had equal probabilities of being selected in the first iteration, promoting a balanced exploration of the solution space.

The probabilities for heuristic selection depend on their performance in previous iterations [14]. To facilitate this, we segment the iterations into groups of 10, and after each segment, we update these probabilities using the formula:

$$\mathcal{W}_{h,(i+1)} = (1-r)\mathcal{W}_{h,i} + r\frac{\pi_{h,i}}{\theta_{h,i}}$$

Here, r represents the *reaction factor*, which determines how much the recent performance change of a heuristic in a segment affects its weight in the next segment. $\pi_{h,i}$ indicates the score that heuristic h earned during segment i, and $\theta_{h,i}$ represents the number of times heuristic h was used during segment i. In our scoring system, a heuristic receives a score of 1 if it improves the current solution, and 0 otherwise.

Additionally, our configuration ensures that no heuristic is entirely excluded from consideration, and we set the minimum probability for any heuristic to be selected at 0.05. We achieve this by normalizing the probabilities so that the minimum value is 0.05.

The adaptive heuristic approach was tested under two distinct neighborhood size configurations: fixed and dynamic. In the dynamic size configuration, aside from the common 300-second time limit (Large) for each iteration, an additional scenario with a 120-second time limit (Small) was explored. This variation aimed to influence the algorithm to favor smaller neighborhoods, potentially improving efficiency. In the following Chapters, we refer to these approaches as the Adaptive heuristic with Fixed Neighborhood Size (AFNS) and the Adaptive heuristic with Dynamic Neighborhood Size (ADNS) small and large.

Chapter 6

Experimental Results

6.1 Experimental Setup

The proposed approach was implemented in Python. Gurobi 10.0.2 was employed to solve sub-problem formulations [6]. The computational experiments in this thesis were run on a 64-bit Windows 10 operating system with 64GB RAM, AMD Ryzen 9 5950X 16-Core Processor (3.40 GHz).

6.2 Results

In the upcoming sections, we will delve into the results of our research. First, in Section 6.2.1, we will showcase the initial solutions derived from the diverse attempts outlined in Section 5.2. Subsequently, in Section 6.2.2, we will present the outcomes of diverse approaches employed to enhance these solutions. Additionally, we will conduct a comparative analysis of different configurations within each approach and highlight the best results achieved in each of these approaches.

In the following sections, whenever we refer to *Improvement*, we are referring to improving with respect to the objective value of the initial solution in percentage that is calculated with the formula 6.1. Similarly, by *Gap* we are referring to the gap from the objective value of the best-found solution in the method to the objective value of the best-known solution in the literature which is calculated by the formula 6.2, where f is the objective function.

$$Improvement = \frac{f(\text{ initial}) - f(\text{ best found})}{f(\text{ initial})} * 100 \tag{6.1}$$

$$Gap = \frac{f(\text{best found}) - f(\text{best known})}{f(\text{best found})} * 100 \tag{6.2}$$

6.2.1 Initial solutions

Among the 45 instances, we were able to find initial solutions for 38 of them. We attempted to solve these instances using the main model with a 24-hour run time, but unfortunately, we could not find feasible solutions for any of them. However, we did manage to obtain initial solutions for 32 instances by not setting an objective function for the model and configuring Gurobi to prioritize finding a feasible solution over a better solution.

When we treated the hard constraints as soft constraints in the objective function, we still could not find feasible solutions for the remaining instances with 24-hour run time. Finally, for the last 6 instances where we obtained initial feasible solutions, we applied our adaptive heuristic algorithm to the model from the previous attempt (the one with hard constraints as soft constraints in the objective function). A complete set of initial objective values obtained for each instance can be found in Table 6.1.

Table 6.1: Complete set of initial objective values obtained

Instance	Initial solution	Instance	Initial solution	Instance	Initial solution
1_1	1762	2_3	18848	3_1	3253
1_2	501	2_4	908	3_2	6419
1_3	10603	2_5	11801	3_3	16214
1_6	16061	2_6	11690	3_4	1084
1_7	11213	2_7	15790	3_6	10973
1_8	7123	2_8	1447	3_7	8441
1_9	14448	2_9	12615	3_8	11885
1_11	11375	2_10	2569	3_9	11614
1_12	13950	2_11	4403	3_11	1141
1_13	1614	2_12	15692	3_12	10240
1_14	15988	2_13	6842	3_13	21622
1_15	20641	2_14	3441	3_14	4597
		2_15	18055	3_15	16575

6.2.2 Improving solutions

In this section, we evaluate the performance of each heuristic combination, as described in Section 5.3.4, using two key metrics mentioned in Section 6.2: Improvement and Gap. Comprehensive results for each instance can be found in Tables A.1 to A.8.

Single heuristic fixed-size

In our experimentation with various single heuristics employing fixed-size neighborhoods for each instance, we observed that the heuristics focusing on slots tend to outperform those focusing on teams. Notably, the heuristic *Random Slots* exhibited the best performance, achieving an average Improvement of 67.88% and a median Improvement of 80.55%. Conversely, the *Costliest Ones Teams* heuristic achieved an average Improvement of 49.81% and a median Improvement of 62.08%. For a more detailed breakdown of each heuristic's performance, refer to Figure 6.1.

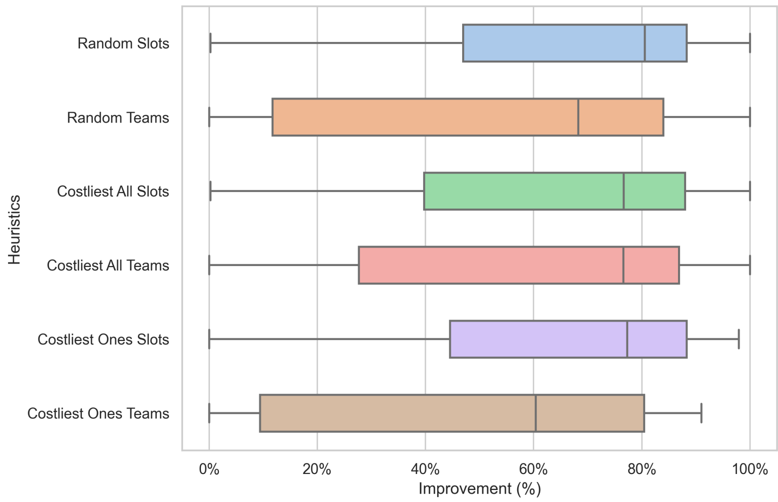

Figure 6.1: Improvement across all instances for single heuristics with fixed-size neighborhoods

When considering the Gap, as depicted in Figure 6.2, the *Random Slots* heuristic stands out as the most effective. It achieves an average Gap of 48.04% and a median Gap of 40.86%. In contrast, the *Costliest Ones Teams* heuristic demonstrates the poorest performance, with an average Gap of 68.91% and a median Gap of 69.96%. This analysis reaffirms the superiority of slot-based heuristics compared to team-based ones in our problem context. It suggests that team-based heuristics may not provide the necessary flexibility for exploring the solution space, leading to their inferior performance.

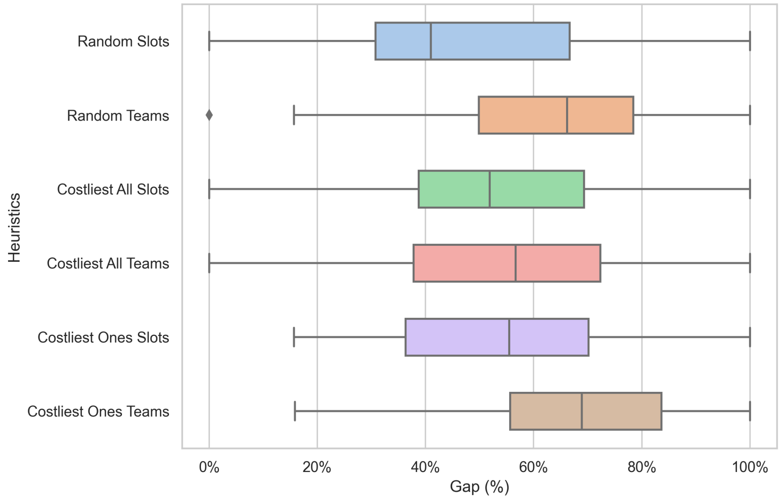

Figure 6.2: Gap across all instances for single heuristics with fixed-size neighborhoods

Single heuristic dynamic-size

When we introduce flexibility regarding the neighborhood sizes, the performance of all six heuristics improves significantly, as shown in Figure 6.3. In this configuration, the *Random Slots* heuristic demonstrates the most remarkable Improvement, achieving an average of 69.56% and a median of 81.62%, while the *Costliest Ones Teams* heuristic exhibits the least Improvement, with an average of 62.14% and a median of 74.94%. A

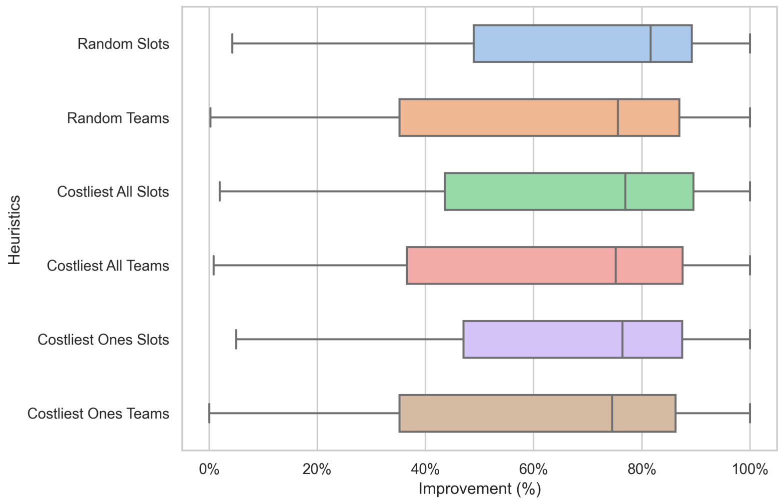

Figure 6.3: Improvement across all instances for single heuristics with dynamic-size neighborhoods

similar trend is observed in Figure 6.4 when considering the Gap. Here again, the *Random*

Slots heuristic outperforms others, with the smallest Gap, averaging 46.41% and a median of 43.66%. Conversely, the *Costliest Ones Teams* heuristic has the largest Gap, with an average of 55.64% and a median of 50.71%. Moreover, it is evident that slot-based heuristics consistently outperform their team-based counterparts. This highlights the value of incorporating flexibility in neighborhood sizes to enhance heuristic performance.

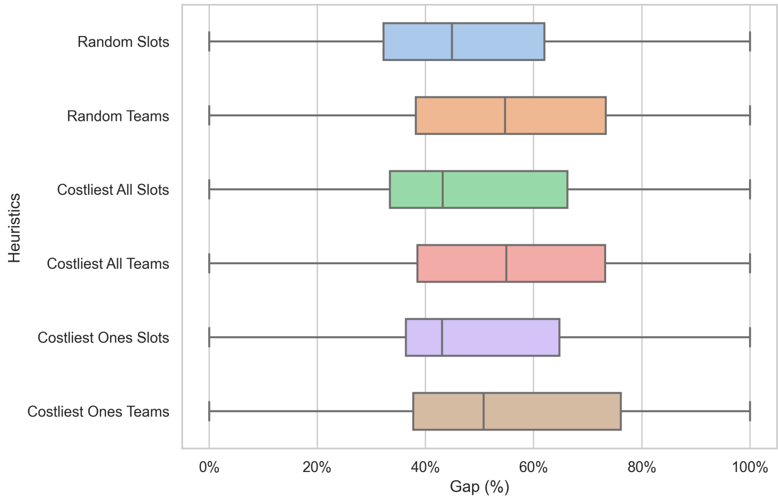

Figure 6.4: Gap across all instances for single heuristics with fixed-size neighborhoods

Double Heuristics Fixed-Size

In this section, we aimed to investigate how combining heuristics of the same category with respect to neighborhood selection influences the final solution. As explained in section 5.3.4, we introduced a new configuration that explores fixed-size large neighborhoods. Figures 6.5 and 6.6 illustrate that combining the *random* heuristics does not outperform the *Random Slots* heuristic alone. However, the combined version of *Costliest Ones* and *Costliest All* heuristics perform better than each of their respective single heuristics. Interestingly, the new heuristics do not yield promising results and exhibit the worst performance among all the double heuristics. This underscores the importance of selecting heuristics carefully, as not all combinations prove equally effective in improving solutions.

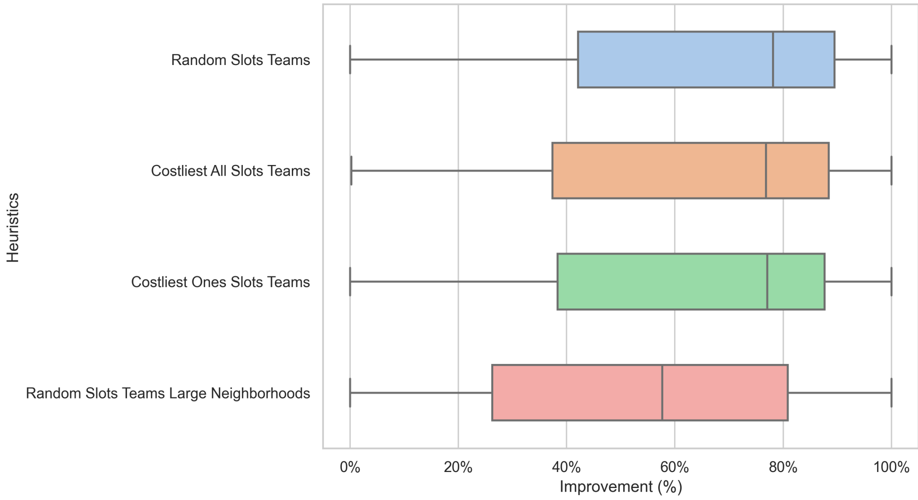

Figure 6.5: Improvement across all instances for double heuristics with dynamic-size neighborhoods

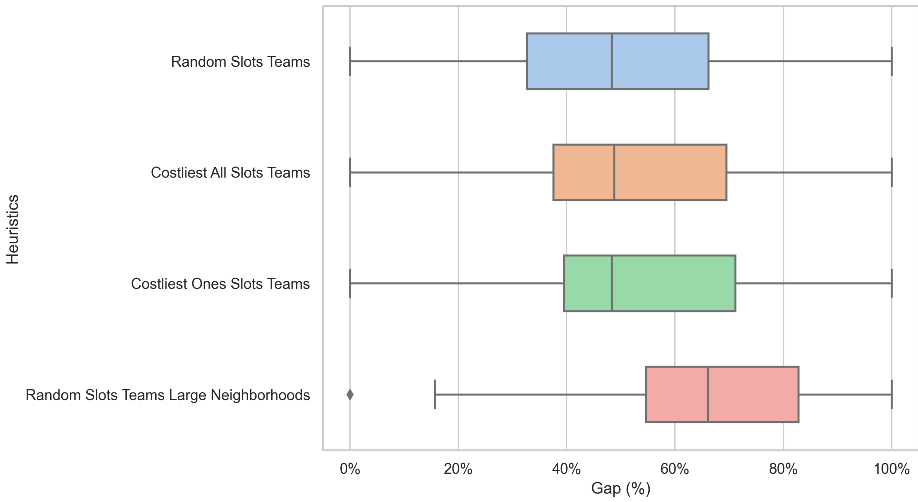

Figure 6.6: Gap across all instances for double heuristics with fixed-size neighborhoods

Adaptive Heuristics

For the adaptive attempts, three different settings were tested, including fixed-size neighborhoods (AFNS) and dynamic-size neighborhoods (ADNS) with shorter (Small) and longer (Large) time limits. The difference in time limits affected the maximum size of the neighborhood that each iteration of experiments could explore.

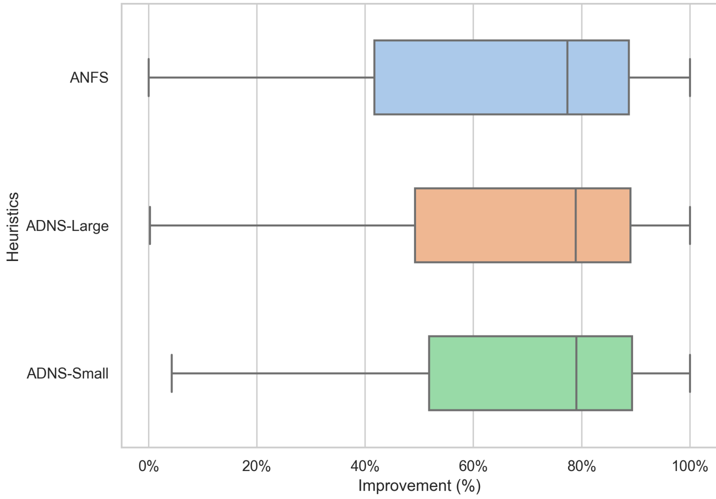

Figure 6.7: Improvement over all instances for adaptive heuristics

Figures 6.7 and 6.8 reveal that, on average, the ADNS-Small approach outperforms both the AFNS and the ADNS-Large. The ADNS-Small consistently achieves higher Improvements and also demonstrates better performance in achieving solutions closer to the best-known solutions (smaller Gap). Specifically, the ADNS-Small achieves an average Improvement of 70.62% with a median of 80.17% and an average Gap of 45.23% with a median of 39.23%. These results highlight the effectiveness of the ADNS-Small approach in solving the problem effectively.

Figure 6.9 presents a comparison of both Improvement and Gap among the best combinations in each of the heuristic combinations. We can see that ADNS outperforms all the other combinations and has the best performance based on both Improvement and Gap and *Random Slots Teams* from double heuristics achieves the poorest results.

Table 6.2 provides the best objective found for each instance among all attempts.

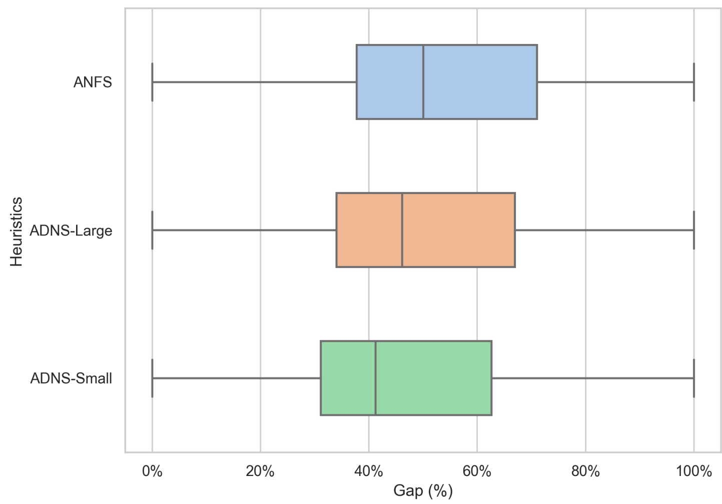

Figure 6.8: Gap across all instances for adaptive heuristics

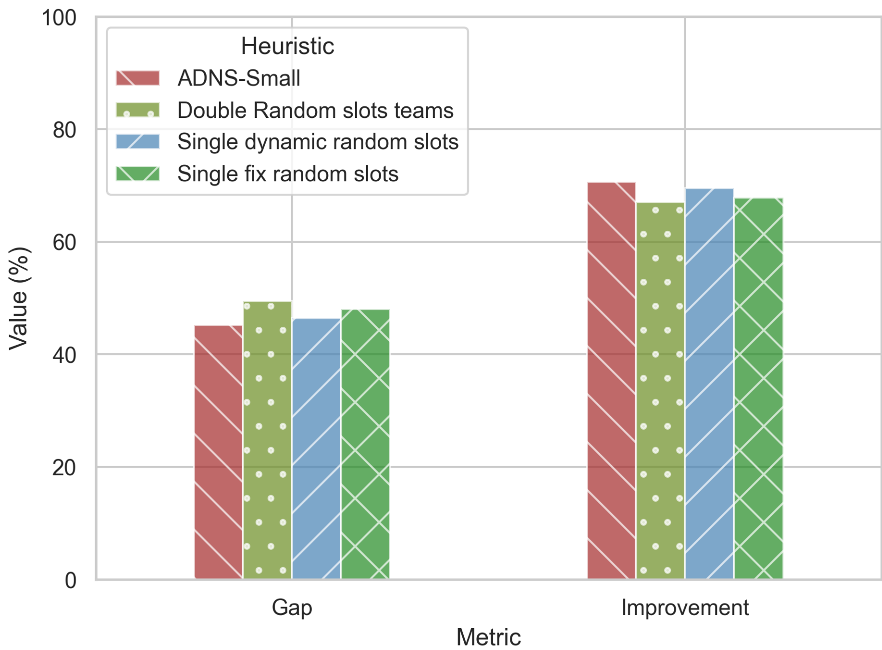

Figure 6.9: Comparison of Gap and Improvement across the best approaches in each combination

Table 6.2: Complete set of best objective values obtained for each instance

Instance	Best found	Instance	Best found	Instance	Best found
1_1	649	2_3	11485	3_1	2343
1_2	346	2_4	11	3_2	6099
1_3	1239	2_5	550	3_3	2952
1_6	4546	2_6	1905	3_4	0
1_7	6839	2_7	2775	3_6	1274
1_8	1496	2_8	234	3_7	2416
1_9	608	2_9	1195	3_8	1267
1_11	6106	2_10	1787	3_9	1313
1_12	1040	2_11	3068	3_11	481
1_13	274	2_12	1150	3_12	5248
1_14	143	2_13	832	3_13	2777
1_15	4737	2_14	1668	3_14	1587
		2_15	1275	3_15	120

Chapter 7

Conclusion and Future Works

In this thesis, we have outlined the problem of generating a schedule for sports tournaments and proposed our solution approach while testing it on the data instances provided by the RobinX project [2]. As with most sports scheduling problems, the process of finding good solutions has been difficult as we consider many conditions regarding competition fairness and criteria requested by stakeholders (e.g., clubs, broadcasters, government), resulting in many and perhaps conflicting constraints. We applied various approaches to find initial solutions and managed this task for 38 out of 45 instances.

We have conducted a comparative analysis of different methods for solving complex sports scheduling problems. In this approach, we employed a fix-and-optimize matheuristic method by creating multiple sub-problems and attempting to solve them instead of the whole problem altogether. We investigated how using an adaptive heuristic method could outperform the single and double heuristic methods. In addition, we explored how having the dynamic-size neighborhoods would affect the performance of our method. Our results demonstrated that in a predefined fixed time limit, among all different configurations, solving multiple small-sized sub-problems outperforms solving large-sized sub-problems. Specifically, our ADNS-Small approach outperformed all the other heuristics examined in this thesis, demonstrating its effectiveness.

Future research in this area could explore several paths to improve the proposed heuristics. One path is to investigate alternative strategies for dynamically adjusting neighborhood sizes, potentially enhancing the overall algorithm's performance. Researchers might also experiment with different scoring systems that influence how heuristics are selected, and explore the use of diverse initial solutions, if possible. Additionally, exploring and combining other existing heuristics from the literature could be beneficial. Another potential direction is to develop and employ an escape heuristic for situations where the algorithm becomes trapped in local optima. This could involve keeping a record of explored neighborhoods to avoid revisiting them when no improvement is observed, thus saving computational resources. Moreover, extending the time limit for experiments may yield more comprehensive results. Lastly, leveraging advanced reinforcement learning techniques could assist in optimizing solution quality and heuristic selection, especially in scenarios with specific constraints for each problem instance.

Bibliography

[1] Dirk Briskorn, Andreas Drexl, and Frits C. R. Spieksma. Round robin tournaments and three index assignments. *4OR*, 8(4):365–374, December 2010. ISSN 1619-4500, 1614-2411. doi: 10.1007/s10288-010-0123-y.
URL: `http://link.springer.com/10.1007/s10288-010-0123-y`.

[2] David Van Bulck, Dries Goossens, Jörn Schönberger, and Mario Guajardo. RobinX: A three-field classification and unified data format for round-robin sports timetabling. *European Journal of Operational Research*, 280(2):568–580, 2020. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2019.07.023.
URL: `https://www.sciencedirect.com/science/article/pii/S0377221719305879`.

[3] Dominique De Werra. Scheduling in sports. *Studies on graphs and discrete programming*, 11:381–395, 1981. Publisher: Amsterdam.

[4] George HG Fonseca and Túlio AM Toffolo. A fix-and-optimize heuristic for the ITC2021 sports timetabling problem. *Journal of Scheduling*, 25(3):273–286, 2022. ISBN: 1094-6136 Publisher: Springer.

[5] Graham Kendall, Sigrid Knust, Celso C. Ribeiro, and Sebastián Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1): 1–19, 2010. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2009.05.013.
URL: `https://www.sciencedirect.com/science/article/pii/S0305054809001543`.

[6] Gurobi Optimization LLC. Gurobi Optimizer Reference Manual, 2023.
URL: `https://www.gurobi.com`.

[7] Vittorio Maniezzo, Marco Antonio Boschetti, and Thomas G. Stützle. *Matheuristics: algorithms and implementations*. EURO advanced tutorials on operational research. Springer, Cham, 2021. ISBN 978-3-030-70279-3 978-3-030-70276-2.

[8] Antony E. Phillips, Michael O'Sullivan, and Cameron Walker. An adaptive large neighbourhood search matheuristic for the ITC2021 Sports Timetabling Competition. In *Proceedings of the 13th International Conference on the Practice and Theory of Automated Timetabling-PATAT*, volume 2, 2021.

[9] Celso C. Ribeiro. Sports scheduling: Problems and applications. *International Transactions in Operational Research*, 19(1-2):201–226, January 2012. ISSN 09696016. doi: 10.1111/j.1475-3995.2011.00819.x.
URL: `https://onlinelibrary.wiley.com/doi/10.1111/j.1475-3995.2011.00819.x`.

[10] Celso C. Ribeiro and Sebastián Urrutia. Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research*, 179(3):775–787, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.03.061.
URL: `https://www.sciencedirect.com/science/article/pii/S0377221705007368`.

[11] Roberto Maria Rosati, Matteo Petris, Luca Di Gaspero, and Andrea Schaerf. Multi-neighborhood simulated annealing for the sports timetabling competition ITC2021. *Journal of Scheduling*, 25(3):301–319, 2022. ISBN: 1094-6136 Publisher: Springer.

[12] Jan A. M. Schreuder. Combinatorial aspects of construction of competition Dutch Professional Football Leagues. *Discrete Applied Mathematics*, 35(3):301–312, 1992. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(92)90252-6.
URL: `https://www.sciencedirect.com/science/article/pii/0166218X92902526`.

[13] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency.* Number 24 in Algorithms and combinatorics. Springer, Berlin ; New York, 2003. ISBN 978-3-540-44389-6.

[14] David Pisinger Stefan Ropke. An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 40(4):455–472, 2006. doi: https://doi.org/10.1287/trsc.1050.0135.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018. ISBN 0-262-35270-2.

[16] Michael A. Trick. Sports Scheduling. In Pascal Van Hentenryck and Michela Milano, editors, *Hybrid Optimization*, volume 45, pages 489–508. Springer New York, New York, NY, 2011. ISBN 978-1-4419-1643-3 978-1-4419-1644-0. doi: 10.1007/978-1-4419-1644-0_15.
URL: `http://link.springer.com/10.1007/978-1-4419-1644-0_15`. Series Title: Springer Optimization and Its Applications.

[17] David Van Bulck. *Sports timetabling: theoretical results and new insights in algorithm performance (PhD dissertation)*. PhD thesis, September 2020.

[18] David Van Bulck and Dries Goossens. First-break-heuristically-schedule: Constructing highly-constrained sports timetables. *Operations Research Letters*, 51(3):326–331, 2023. ISBN: 0167-6377 Publisher: Elsevier.

[19] David Van Bulck, Dries Goossens, Jeroen Belien, and Morteza Davari. The fifth international timetabling competition (itc 2021): Sports timetabling. In *MathSport international 2021*, pages 117–122. University of Reading, 2021.

[20] David Van Bulck, Dries Goossens, Jan-Patrick Clarner, Angelos Dimitsas, George H. G. Fonseca, Carlos Lamas-Fernandez, Martin Mariusz Lester, Jaap Pedersen, Antony E. Phillips, and Roberto Maria Rosati. Which algorithm to select in sports timetabling? 2023. doi: 10.48550/ARXIV.2309.03229.
URL: `https://arxiv.org/abs/2309.03229`. Publisher: arXiv Version Number: 1.

Appendix A

Results

Single heuristic

Table A.1: Single fixed-size heuristics Improvement over Initial sols

		Rand. W.		Rand. T.		costliest all W.		costliest all T.		costliest ones W.		costliest ones T.	
Instance	Init. Sol.	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%
1_1	1762	794	54.94%	1640	6.92%	1598	9.31%	1683	4.48%	1475	16.29%	1723	2.21%
1_2	501	467	6.79%	501	0.00%	467	6.79%	501	0.00%	501	0.00%	501	0.00%
1_3	10603	1328	87.48%	1938	81.72%	1593	84.98%	1424	86.57%	1563	85.26%	2364	77.70%
1_6	16061	4875	69.65%	5660	64.76%	5660	64.76%	5660	64.76%	5660	64.76%	5660	64.76%
1_7	11213	6839	39.01%	10833	3.39%	8004	28.62%	10803	3.66%	7180	35.97%	10803	3.66%
1_8	7123	1531	78.51%	2001	71.91%	1553	78.20%	5123	28.08%	1596	77.59%	2355	66.94%
1_9	14448	792	94.52%	1182	91.82%	783	94.58%	763	94.72%	848	94.13%	1587	89.02%
1_11	11375	7470	34.33%	10480	7.87%	7292	35.89%	10370	8.84%	6326	44.39%	10560	7.16%
1_12	13950	1975	85.84%	1940	86.09%	1650	88.17%	1820	86.95%	1365	90.22%	2075	85.13%
1_13	1614	274	83.02%	795	50.74%	336	79.18%	417	74.16%	342	78.81%	1032	36.06%
1_14	15988	429	97.32%	1040	93.50%	406	97.46%	568	96.45%	569	96.44%	1532	90.42%
1_15	20641	4737	77.05%	6359	69.19%	5703	72.37%	4832	76.59%	6629	67.88%	7077	65.71%
2_3	18848	11970	36.49%	11970	36.49%	11995	36.36%	11678	38.04%	11970	36.49%	11970	36.49%
2_4	908	25	97.25%	81	91.08%	33	96.37%	15	98.35%	64	92.95%	121	86.67%
2_5	11801	777	93.42%	1346	88.59%	889	92.47%	972	91.76%	886	92.49%	2396	79.70%
2_6	11690	2275	80.54%	3315	71.64%	2530	78.36%	2585	77.89%	2650	77.33%	3325	71.56%
2_7	15790	3068	80.57%	5688	63.98%	3706	76.53%	2960	81.25%	3876	75.45%	5725	63.74%
2_8	1447	255	82.38%	459	68.28%	341	76.43%	335	76.85%	315	78.23%	598	58.67%
2_9	12615	1265	89.97%	1915	84.82%	1420	88.74%	1665	86.80%	1365	89.18%	2370	81.21%
2_10	2569	2014	21.60%	2443	4.90%	2171	15.49%	2464	4.09%	1956	23.86%	2449	4.67%
2_11	4403	3068	30.32%	4023	8.63%	4108	6.70%	3737	15.13%	3853	12.49%	4218	4.20%

2_12	15692	1413	91.00%	2506	84.03%	1468	90.64%	1668	89.37%	2046	86.96%	4067	74.08%
2_13	6842	862	87.40%	2420	64.63%	833	87.83%	946	86.17%	952	86.09%	4476	34.58%
2_14	3441	1780	48.27%	3142	8.69%	2023	41.21%	3116	9.44%	1904	44.67%	3145	8.60%
2_15	18055	1434	92.06%	1975	89.06%	1406	92.21%	2799	84.50%	1299	92.81%	2147	88.11%
3_1	3253	2708	16.75%	3106	4.52%	3110	4.40%	2364	27.33%	3214	1.20%	3168	2.61%
3_2	6419	6404	0.23%	6404	0.23%	6404	0.23%	6404	0.23%	6404	0.23%	6419	0.00%
3_3	16214	2983	81.60%	4834	70.19%	3328	79.47%	3829	76.38%	3664	77.40%	6529	59.73%
3_4	1084	0	100.00%	0	100.00%	0	100.00%	0	100.00%	22	97.97%	144	86.72%
3_6	10973	1428	86.99%	1917	82.53%	1480	86.51%	1422	87.04%	1370	87.51%	1940	82.32%
3_7	8441	2416	71.38%	4975	41.06%	3207	62.01%	2981	64.68%	2551	69.78%	5585	33.83%
3_8	11885	1291	89.14%	1902	84.00%	1460	87.72%	1494	87.43%	1294	89.11%	1963	83.48%
3_9	11614	4810	58.58%	2324	79.99%	1704	85.33%	1560	86.57%	1604	86.19%	2599	77.62%
3_11	1141	620	45.66%	1056	7.45%	676	40.75%	1061	7.01%	481	57.84%	1081	5.26%
3_12	10240	5807	43.29%	8730	14.75%	6267	38.80%	6089	40.54%	6075	40.67%	9190	10.25%
3_13	21622	2777	87.16%	6319	70.78%	5048	76.65%	4227	80.45%	5550	74.33%	8560	60.41%
3_14	4597	1745	62.04%	2711	41.03%	1705	62.91%	1886	58.97%	1871	59.30%	3749	18.45%
3_15	16575	540	96.74%	1145	93.09%	560	96.62%	480	97.10%	540	96.74%	1485	91.04%
Average			67.88		54.80		64.24		59.96		65.24		49.81

Table A.2: Single fixed-size heuristics Gap to Best knowns

		Rand. W.		Rand. T.		costliest all W.		costliest all T.		costliest ones W.		costliest ones T.	
Instance	BK	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%
1_1	362	794	54.41	1640	77.93	1598	77.35	1683	78.49	1475	75.46	1723	78.99
1_2	145	467	68.95	501	71.06	467	68.95	501	71.06	501	71.06	501	71.06
1_3	992	1328	25.30	1938	48.81	1593	37.73	1424	30.34	1563	36.53	2364	58.04
1_6	3325	4875	31.79	5660	41.25	5660	41.25	5660	41.25	5660	41.25	5660	41.25
1_7	4763	6839	30.36	10833	56.03	8004	40.49	10803	55.91	7180	33.66	10803	55.91
1_8	1051	1531	31.35	2001	47.48	1553	32.32	5123	79.48	1596	34.15	2355	55.37
1_9	56	792	92.93	1182	95.26	783	92.85	763	92.66	848	93.40	1587	96.47
1_11	4426	7470	40.75	10480	57.77	7292	39.30	10370	57.32	6326	30.03	10560	58.09
1_12	315	1975	84.05	1940	83.76	1650	80.91	1820	82.69	1365	76.92	2075	84.82
1_13	121	274	55.84	795	84.78	336	63.99	417	70.98	342	64.62	1032	88.28
1_14	4	429	99.07	1040	99.62	406	99.01	568	99.30	569	99.30	1532	99.74
1_15	3362	4737	29.03	6359	47.13	5703	41.05	4832	30.42	6629	49.28	7077	52.49
2_3	9542	11970	20.28	11970	20.28	11995	20.45	11678	18.29	11970	20.28	11970	20.28
2_4	7	25	72.00	81	91.36	33	78.79	15	53.33	64	89.06	121	94.21
2_5	279	777	64.09	1346	79.27	889	68.62	972	71.30	886	68.51	2396	88.36
2_6	1120	2275	50.77	3315	66.21	2530	55.73	2585	56.67	2650	57.74	3325	66.32
2_7	1783	3068	41.88	5688	68.65	3706	51.89	2960	39.76	3876	54.00	5725	68.86
2_8	129	255	49.41	459	71.90	341	62.17	335	61.49	315	59.05	598	78.43
2_9	415	1265	67.19	1915	78.33	1420	70.77	1665	75.08	1365	69.60	2370	82.49
2_10	1250	2014	37.93	2443	48.83	2171	42.42	2464	49.27	1956	36.09	2449	48.96
2_11	2446	3068	20.27	4023	39.20	4108	40.46	3737	34.55	3853	36.52	4218	42.01

2_12	599	1413	57.61	2506	76.10	1468	59.20	1668	64.09	2046	70.72	4067	85.27
2_13	252	862	70.77	2420	89.59	833	69.75	946	73.36	952	73.53	4476	94.37
2_14	1140	1780	35.96	3142	63.72	2023	43.65	3116	63.41	1904	40.13	3145	63.75
2_15	485	1434	66.18	1975	75.44	1406	65.50	2799	82.67	1299	62.66	2147	77.41
3_1	1922	2708	29.03	3106	38.12	3110	38.20	2364	18.70	3214	40.20	3168	39.33
3_2	5400	6404	15.68	6404	15.68	6404	15.68	6404	15.68	6404	15.68	6419	15.87
3_3	2369	2983	20.58	4834	50.99	3328	28.82	3829	38.13	3664	35.34	6529	63.72
3_4	0	0	0.00	0	0.00	0	0.00	0	0.00	22	100.00	144	100.00
3_6	923	1428	35.36	1917	51.85	1480	37.64	1422	35.09	1370	32.63	1940	52.42
3_7	1558	2416	35.51	4975	68.68	3207	51.42	2981	47.74	2551	38.93	5585	72.10
3_8	934	1291	27.65	1902	50.89	1460	36.03	1494	37.48	1294	27.82	1963	52.42
3_9	498	4810	89.65	2324	78.57	1704	70.77	1560	68.08	1604	68.95	2599	80.84
3_11	202	620	67.42	1056	80.87	676	70.12	1061	80.96	481	58.00	1081	81.31
3_12	3428	5807	40.97	8730	60.73	6267	45.30	6089	43.70	6075	43.57	9190	62.70
3_13	1820	2777	34.46	6319	71.20	5048	63.95	4227	56.94	5550	67.21	8560	78.74
3_14	1202	1745	31.12	2711	55.66	1705	29.50	1886	36.27	1871	35.76	3749	67.94
3_15	0	540	100.00	1145	100.00	560	100.00	480	100.00	540	100.00	1485	100.00
Average			48.04		63.24		53.47		55.58		55.46		68.91

Single Heuristic dynamic size

Table A.3: Single dynamic-size heuristics Improvement over Initial sols

		Rand. W.		Rand. T.		costliest all W.		costliest all T.		costliest ones W.		costliest ones T.	
Instance	Init. Sol.	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%
1_1	1762	657	62.71	1447	17.88	806	54.26	1048	40.52	736	58.23	1600	9.19
1_2	501	390	22.16	479	4.39	379	24.35	486	2.99	397	20.76	501	0.00
1_3	10603	1439	86.43	1542	85.46	1239	88.31	1539	85.49	1539	85.49	1528	85.59
1_6	16061	4546	71.70	5599	65.14	5660	64.76	5660	64.76	5660	64.76	5383	66.48
1_7	11213	7278	35.09	8345	25.58	7925	29.32	8058	28.14	7543	32.73	8858	21.00
1_8	7123	1751	75.42	1526	78.58	1849	74.04	1534	78.46	1680	76.41	1510	78.80
1_9	14448	633	95.62	768	94.68	697	95.18	808	94.41	4811	66.70	913	93.68
1_11	11375	6929	39.09	7239	36.36	7529	33.81	7576	33.40	6106	46.32	7524	33.85
1_12	13950	1095	92.15	1721	87.66	1110	92.04	1690	87.89	1100	92.11	1630	88.32
1_13	1614	482	70.14	517	67.97	401	75.15	497	69.21	403	75.03	527	67.35
1_14	15988	164	98.97	166	98.96	247	98.46	143	99.11	389	97.57	165	98.97
1_15	20641	4927	76.13	5389	73.89	5485	73.43	5457	73.56	5695	72.41	5741	72.19
2_3	18848	11970	36.49	11970	36.49	11485	39.07	11676	38.05	11970	36.49	11819	37.29
2_4	908	17	98.13	36	96.04	23	97.47	35	96.15	11	98.79	42	95.37
2_5	11801	550	95.34	1067	90.96	642	94.56	1061	91.01	649	94.50	1128	90.44
2_6	11690	2150	81.61	2850	75.62	1930	83.49	3085	73.61	2080	82.21	2975	74.55
2_7	15790	2775	82.43	3688	76.64	3500	77.83	3453	78.13	3647	76.90	3173	79.91
2_8	1447	241	83.34	368	74.57	251	82.65	336	76.78	282	80.51	357	75.33
2_9	12615	1295	89.73	1435	88.62	1260	90.01	1560	87.63	1395	88.94	3668	70.92

2_10	2569	1872	27.13	2300	10.47	1865	27.40	2333	9.19	2052	20.12	2341	8.88
2_11	4403	3308	24.87	3588	18.51	3698	16.01	3588	18.51	3833	12.95	3588	18.51
2_12	15692	1150	92.67	2100	86.62	1297	91.73	1974	87.42	1279	91.85	1379	91.21
2_13	6842	832	87.84	1422	79.22	862	87.40	1700	75.15	922	86.52	1652	75.86
2_14	3441	1914	44.38	2270	34.03	1995	42.02	2233	35.11	1876	45.48	2183	36.56
2_15	18055	1538	91.48	1782	90.13	1500	91.69	1800	90.03	1478	91.81	1652	90.85
3_1	3253	2344	27.94	2429	25.33	2662	18.17	2481	23.73	3059	5.96	2586	20.50
3_2	6419	6145	4.27	6404	0.23	6294	1.95	6364	0.86	6099	4.99	6404	0.23
3_3	16214	2978	81.63	3678	77.32	3359	79.28	3528	78.24	2952	81.79	3429	78.85
3_4	1084	0	100.00	0	100.00	0	100.00	0	100.00	0	100.00	0	100.00
3_6	10973	1372	87.50	1397	87.27	1376	87.46	1337	87.82	1450	86.79	1435	86.92
3_7	8441	3029	64.12	3439	59.26	2592	69.29	3039	64.00	2736	67.59	3164	62.52
3_8	11885	1329	88.82	1646	86.15	1294	89.11	1647	86.14	1407	88.16	1739	85.37
3_9	11614	1313	88.69	1659	85.72	1440	87.60	1625	86.01	1560	86.57	1814	84.38
3_11	1141	531	53.46	961	15.78	696	39.00	1031	9.64	596	47.77	956	16.21
3_12	10240	5954	41.86	7064	31.02	5619	45.13	7705	24.76	6006	41.35	6948	32.15
3_13	21622	3464	83.98	4907	77.31	4988	76.93	5010	76.83	4786	77.87	5229	75.82
3_14	4597	1787	61.13	1982	56.88	1669	63.69	1959	57.39	1779	61.30	1916	58.32
3_15	16575	160	99.03	150	99.10	120	99.28	180	98.91	140	99.16	180	98.91
Average			69.56		63.05		67.93		63.40		66.97		62.14

Table A.4: Single dynamic-size heuristics Gap to Best knowns

		Rand. W.		Rand. T.		costliest all W.		costliest all T.		costliest ones W.		costliest ones T.	
Instance	BK	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%
1_1	362	657	44.90	1447	74.98	806	55.09	1048	65.46	736	50.82	1600	77.38
1_2	145	390	62.82	479	69.73	379	61.74	486	70.16	397	63.48	501	71.06
1_3	992	1439	31.06	1542	35.67	1239	19.94	1539	35.54	1539	35.54	1528	35.08
1_6	3325	4546	26.86	5599	40.61	5660	41.25	5660	41.25	5660	41.25	5383	38.23
1_7	4763	7278	34.56	8345	42.92	7925	39.90	8058	40.89	7543	36.86	8858	46.23
1_8	1051	1751	39.98	1526	31.13	1849	43.16	1534	31.49	1680	37.44	1510	30.40
1_9	56	633	91.15	768	92.71	697	91.97	808	93.07	4811	98.84	913	93.87
1_11	4426	6929	36.12	7239	38.86	7529	41.21	7576	41.58	6106	27.51	7524	41.17
1_12	315	1095	71.23	1721	81.70	1110	71.62	1690	81.36	1100	71.36	1630	80.67
1_13	121	482	74.90	517	76.60	401	69.83	497	75.65	403	69.98	527	77.04
1_14	4	164	97.56	166	97.59	247	98.38	143	97.20	389	98.97	165	97.58
1_15	3362	4927	31.76	5389	37.61	5485	38.71	5457	38.39	5695	40.97	5741	41.44
2_3	9542	11970	20.28	11970	20.28	11485	16.92	11676	18.28	11970	20.28	11819	19.27
2_4	7	17	58.82	36	80.56	23	69.57	35	80.00	11	36.36	42	83.33
2_5	279	550	49.27	1067	73.85	642	56.54	1061	73.70	649	57.01	1128	75.27
2_6	1120	2150	47.91	2850	60.70	1930	41.97	3085	63.70	2080	46.15	2975	62.35
2_7	1783	2775	35.75	3688	51.65	3500	49.06	3453	48.36	3647	51.11	3173	43.81
2_8	129	241	46.47	368	64.95	251	48.61	336	61.61	282	54.26	357	63.87
2_9	415	1295	67.95	1435	71.08	1260	67.06	1560	73.40	1395	70.25	3668	88.69
2_10	1250	1872	33.23	2300	45.65	1865	32.98	2333	46.42	2052	39.08	2341	46.60
2_11	2446	3308	26.06	3588	31.83	3698	33.86	3588	31.83	3833	36.19	3588	31.83

2_12	599	1150	47.91	2100	71.48	1297	53.82	1974	69.66	1279	53.17	1379	56.56
2_13	252	832	69.71	1422	82.28	862	70.77	1700	85.18	922	72.67	1652	84.75
2_14	1140	1914	40.44	2270	49.78	1995	42.86	2233	48.95	1876	39.23	2183	47.78
2_15	485	1538	68.47	1782	72.78	1500	67.67	1800	73.06	1478	67.19	1652	70.64
3_1	1922	2344	18.00	2429	20.87	2662	27.80	2481	22.53	3059	37.17	2586	25.68
3_2	5400	6145	12.12	6404	15.68	6294	14.20	6364	15.15	6099	11.46	6404	15.68
3_3	2369	2978	20.45	3678	35.59	3359	29.47	3528	32.85	2952	19.75	3429	30.91
3_4	0	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
3_6	923	1372	32.73	1397	33.93	1376	32.92	1337	30.96	1450	36.34	1435	35.68
3_7	1558	3029	48.56	3439	54.70	2592	39.89	3039	48.73	2736	43.06	3164	50.76
3_8	934	1329	29.72	1646	43.26	1294	27.82	1647	43.29	1407	33.62	1739	46.29
3_9	498	1313	62.07	1659	69.98	1440	65.42	1625	69.35	1560	68.08	1814	72.55
3_11	202	531	61.96	961	78.98	696	70.98	1031	80.41	596	66.11	956	78.87
3_12	3428	5954	42.43	7064	51.47	5619	38.99	7705	55.51	6006	42.92	6948	50.66
3_13	1820	3464	47.46	4907	62.91	4988	63.51	5010	63.67	4786	61.97	5229	65.19
3_14	1202	1787	32.74	1982	39.35	1669	27.98	1959	38.64	1779	32.43	1916	37.27
3_15	0	160	100.00	150	100.00	120	100.00	180	100.00	140	100.00	180	100.00
Average			46.41		55.36		49.04		54.93		49.18		55.64

Both

Table A.5: Double heuristics Improvement over Initial sols

		Rand. W.T.		costliest all W.T.		costliest ones W.T.		Rand. L.W.T	
Instance	Init. Sol.	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%	Obj	Imp.%
1_1	1762	1033	41.37	1626	7.72	1055	40.12	1026	41.77
1_2	501	501	0.00	467	6.79	501	0.00	464	7.39
1_3	10603	1473	86.11	1606	84.85	1598	84.93	2155	79.68
1_6	16061	5165	67.84	5660	64.76	5660	64.76	5695	64.54
1_7	11213	7686	31.45	8370	25.35	7980	28.83	10508	6.29
1_8	7123	1539	78.39	1649	76.85	1526	78.58	3329	53.26
1_9	14448	698	95.17	733	94.93	858	94.06	1562	89.19
1_11	11375	6495	42.90	7901	30.54	7373	35.18	10735	5.63
1_12	13950	1085	92.22	1280	90.82	1290	90.75	2020	85.52
1_13	1614	417	74.16	401	75.15	390	75.84	884	45.23
1_14	15988	468	97.07	505	96.84	626	96.08	267	98.33
1_15	20641	4993	75.81	5482	73.44	6122	70.34	7406	64.12
2_3	18848	11970	36.49	11970	36.49	11970	36.49	11970	36.49
2_4	908	21	97.69	37	95.93	68	92.51	43	95.26
2_5	11801	848	92.81	903	92.35	961	91.86	1584	86.58
2_6	11690	2605	77.72	2550	78.19	2680	77.07	3325	71.56
2_7	15790	3450	78.15	3599	77.21	3214	79.65	5263	66.67
2_8	1447	270	81.34	252	82.58	285	80.30	767	46.99
2_9	12615	1195	90.53	1430	88.66	1610	87.24	2295	81.81
2_10	2569	1877	26.94	2035	20.79	2141	16.66	2349	8.56
2_11	4403	3308	24.87	3603	18.17	4053	7.95	4403	0.00
2_12	15692	1278	91.86	1717	89.06	1548	90.14	3899	75.15
2_13	6842	1022	85.06	1100	83.92	1269	81.45	2898	57.64
2_14	3441	2038	40.77	2057	40.22	1990	42.17	3084	10.37
2_15	18055	1412	92.18	1279	92.92	1465	91.89	3277	81.85
3_1	3253	2950	9.31	3047	6.33	3171	2.52	2914	10.42
3_2	6419	6404	0.23	6404	0.23	6404	0.23	6404	0.23
3_3	16214	3144	80.61	3103	80.86	3623	77.66	7294	55.01
3_4	1084	0	100.00	0	100.00	0	100.00	0	100.00
3_6	10973	1456	86.73	1365	87.56	1554	85.84	1617	85.26
3_7	8441	2591	69.30	2785	67.01	2455	70.92	4894	42.02

3_8	11885	1372	88.46	1407	88.16	1412	88.12	2388	79.91
3_9	11614	1445	87.56	1430	87.69	1729	85.11	2428	79.09
3_11	1141	711	37.69	766	32.87	816	28.48	1131	0.88
3_12	10240	5808	43.28	6325	38.23	6635	35.21	8325	18.70
3_13	21622	3618	83.27	5902	72.70	6621	69.38	4804	77.78
3_14	4597	1587	65.48	1836	60.06	1848	59.80	3042	33.83
3_15	16575	540	96.74	565	96.59	600	96.38	305	98.16
Average			67.04		64.29		64.07		53.72

Table A.6: Double heuristics Gap to Best knowns

		Rand. W.T.		costliest all W.T.		costliest ones W.T.		Rand. L.W.T	
Instance	BK	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%	Obj	Gap.%
1_1	362	1033	64.96	1626	77.74	1055	65.69	1026	64.72
1_2	145	501	71.06	467	68.95	501	71.06	464	68.75
1_3	992	1473	32.65	1606	38.23	1598	37.92	2155	53.97
1_6	3325	5165	35.62	5660	41.25	5660	41.25	5695	41.62
1_7	4763	7686	38.03	8370	43.09	7980	40.31	10508	54.67
1_8	1051	1539	31.71	1649	36.26	1526	31.13	3329	68.43
1_9	56	698	91.98	733	92.36	858	93.47	1562	96.41
1_11	4426	6495	31.86	7901	43.98	7373	39.97	10735	58.77
1_12	315	1085	70.97	1280	75.39	1290	75.58	2020	84.41
1_13	121	417	70.98	401	69.83	390	68.97	884	86.31
1_14	4	468	99.15	505	99.21	626	99.36	267	98.50
1_15	3362	4993	32.67	5482	38.67	6122	45.08	7406	54.60
2_3	9542	11970	20.28	11970	20.28	11970	20.28	11970	20.28
2_4	7	21	66.67	37	81.08	68	89.71	43	83.72
2_5	279	848	67.10	903	69.10	961	70.97	1584	82.39
2_6	1120	2605	57.01	2550	56.08	2680	58.21	3325	66.32
2_7	1783	3450	48.32	3599	50.46	3214	44.52	5263	66.12
2_8	129	270	52.22	252	48.81	285	54.74	767	83.18
2_9	415	1195	65.27	1430	70.98	1610	74.22	2295	81.92
2_10	1250	1877	33.40	2035	38.57	2141	41.62	2349	46.79
2_11	2446	3308	26.06	3603	32.11	4053	39.65	4403	44.45
2_12	599	1278	53.13	1717	65.11	1548	61.30	3899	84.64
2_13	252	1022	75.34	1100	77.09	1269	80.14	2898	91.30
2_14	1140	2038	44.06	2057	44.58	1990	42.71	3084	63.04

2_15	485	1412	65.65	1279	62.08	1465	66.89	3277	85.20
3_1	1922	2950	34.85	3047	36.92	3171	39.39	2914	34.04
3_2	5400	6404	15.68	6404	15.68	6404	15.68	6404	15.68
3_3	2369	3144	24.65	3103	23.65	3623	34.61	7294	67.52
3_4	0	0	0.00	0	0.00	0	0.00	0	0.00
3_6	923	1456	36.61	1365	32.38	1554	40.60	1617	42.92
3_7	1558	2591	39.87	2785	44.06	2455	36.54	4894	68.17
3_8	934	1372	31.92	1407	33.62	1412	33.85	2388	60.89
3_9	498	1445	65.54	1430	65.17	1729	71.20	2428	79.49
3_11	202	711	71.59	766	73.63	816	75.25	1131	82.14
3_12	3428	5808	40.98	6325	45.80	6635	48.33	8325	58.82
3_13	1820	3618	49.70	5902	69.16	6621	72.51	4804	62.11
3_14	1202	1587	24.26	1836	34.53	1848	34.96	3042	60.49
3_15	0	540	100.00	565	100.00	600	100.00	305	100.00
Average			49.52		53.05		54.15		64.81

Adaptive

Table A.7: Adaptive heuristics Improvement over Initial sols

		AFNS		ADNS-Large		ADNS-Small	
Inst.	Init. Sol.	Obj	Imp.%	Obj	Imp.%	Obj	Imp%
1_1	1762	1335	24.23	653	62.94	649	63.17
1_2	501	501	0.00	440	12.18	346	30.94
1_3	10603	1593	84.98	1370	87.08	1284	87.89
1_6	16061	5660	64.76	4864	69.72	4681	70.85
1_7	11213	8044	28.26	7192	35.86	7042	37.20
1_8	7123	1664	76.64	1640	76.98	1496	79.00
1_9	14448	773	94.65	642	95.56	608	95.79
1_11	11375	7003	38.44	6831	39.95	6905	39.30
1_12	13950	1140	91.83	1050	92.47	1040	92.54
1_13	1614	376	76.70	415	74.29	415	74.29
1_14	15988	530	96.69	434	97.29	164	98.97
1_15	20641	5827	71.77	5210	74.76	5075	75.41
2_3	18848	11970	36.49	11970	36.49	11970	36.49
2_4	908	36	96.04	15	98.35	13	98.57
2_5	11801	1091	90.76	678	94.25	750	93.64
2_6	11690	2350	79.90	2400	79.47	1905	83.70
2_7	15790	3570	77.39	3327	78.93	2945	81.35
2_8	1447	235	83.76	289	80.03	234	83.83
2_9	12615	1505	88.07	1300	89.69	1260	90.01
2_10	2569	2008	21.84	1918	25.34	1787	30.44
2_11	4403	3558	19.19	3333	24.30	3103	29.53
2_12	15692	1587	89.89	1303	91.70	1220	92.23
2_13	6842	917	86.60	1208	82.34	1090	84.07
2_14	3441	1971	42.72	2107	38.77	1668	51.53
2_15	18055	1275	92.94	1468	91.87	1467	91.87
3_1	3253	3116	4.21	2718	16.45	2343	27.97
3_2	6419	6404	0.23	6404	0.23	6144	4.28
3_3	16214	3484	78.51	3224	80.12	3499	78.42
3_4	1084	0	100.00	0	100.00	0	100.00
3_6	10973	1362	87.59	1274	88.39	1374	87.48
3_7	8441	2779	67.08	2892	65.74	2554	69.74
3_8	11885	1267	89.34	1415	88.09	1356	88.59
3_9	11614	1639	85.89	1587	86.34	1333	88.52
3_11	1141	596	47.77	546	52.15	546	52.15
3_12	10240	6076	40.66	5498	46.31	5248	48.75
3_13	21622	4900	77.34	3459	84.00	3567	83.50
3_14	4597	1810	60.63	1843	59.91	1721	62.56
3_15	16575	580	96.50	200	98.79	140	99.16
Average			65.53		68.34		70.62

Table A.8: Adaptive heuristics Gap to Best knowns

		AFNS		ADNS-Large		ADNS-Small	
Inst.	BK	Obj	Gap%	Obj	Gap%	Obj	Gap%
1_1	362	1335	72.88	653	44.56	649	44.22
1_2	145	501	71.06	440	67.05	346	58.09
1_3	992	1593	37.73	1370	27.59	1284	22.74
1_6	3325	5660	41.25	4864	31.64	4681	28.97
1_7	4763	8044	40.79	7192	33.77	7042	32.36
1_8	1051	1664	36.84	1640	35.91	1496	29.75
1_9	56	773	92.76	642	91.28	608	90.79
1_11	4426	7003	36.80	6831	35.21	6905	35.90
1_12	315	1140	72.37	1050	70.00	1040	69.71
1_13	121	376	67.82	415	70.84	415	70.84
1_14	4	530	99.25	434	99.08	164	97.56
1_15	3362	5827	42.30	5210	35.47	5075	33.75
2_3	9542	11970	20.28	11970	20.28	11970	20.28
2_4	7	36	80.56	15	53.33	13	46.15
2_5	279	1091	74.43	678	58.85	750	62.80
2_6	1120	2350	52.34	2400	53.33	1905	41.21
2_7	1783	3570	50.06	3327	46.41	2945	39.46
2_8	129	235	45.11	289	55.36	234	44.87
2_9	415	1505	72.43	1300	68.08	1260	67.06
2_10	1250	2008	37.75	1918	34.83	1787	30.05
2_11	2446	3558	31.25	3333	26.61	3103	21.17
2_12	599	1587	62.26	1303	54.03	1220	50.90
2_13	252	917	72.52	1208	79.14	1090	76.88
2_14	1140	1971	42.16	2107	45.89	1668	31.65
2_15	485	1275	61.96	1468	66.96	1467	66.94
3_1	1922	3116	38.32	2718	29.29	2343	17.97
3_2	5400	6404	15.68	6404	15.68	6144	12.11
3_3	2369	3484	32.00	3224	26.52	3499	32.29
3_4	0	0	0.00	0	0.00	0	0.00
3_6	923	1362	32.23	1274	27.55	1374	32.82
3_7	1558	2779	43.94	2892	46.13	2554	39.00
3_8	934	1267	26.28	1415	33.99	1356	31.12
3_9	498	1639	69.62	1587	68.62	1333	62.64
3_11	202	596	66.11	546	63.00	546	63.00
3_12	3428	6076	43.58	5498	37.65	5248	34.68
3_13	1820	4900	62.86	3459	47.38	3567	48.98
3_14	1202	1810	33.59	1843	34.78	1721	30.16
3_15	0	580	100.00	200	100.00	140	100.00
Average			53.08		48.69		45.53