Universidade Federal de Minas Gerais
Graduate Program in Electrical Engineering

Formulations and Algorithms for Timetabling

George Henrique Godim da Fonseca

Thesis submitted to the examination board designated by the Graduate Program in Electrical Engineering collegiate as partial requirement to obtain a Ph.D. in Electrical Engineering from the Universidade Federal de Minas Gerais.

Advisor: Prof. Eduardo G. Carrano, Ph.D.
Co-advisor: Prof. Haroldo G. Santos, Ph.D.

Belo Horizonte, April 2017

Resumo

O Problema de Programação de Horários Educacionais consiste em alocar horários e recursos a eventos respeitando diversas restrições. Além de sua importância prática, esse problema é classificado como $\mathcal{NP}$-Difícil na maioria de suas formulações. Essa tese considera o formato XHSTT devido à sua capacidade de lidar com diversas particularidades da programação de horários e à sua relevância na literatura recente. O principal objetivo desse trabalho é desenvolver novas formulações e algoritmos para esse problema. A respeito de formulações, uma formulação alternativa e uma abordagem de geração de colunas foram propostas. Na área de algoritmos, duas metaheurísticas e algumas variações delas foram desenvolvidas. Uma heurística matemática de Fixa-e-Otimiza específica para o problema também foi proposta, junto com uma variação nomeada Fixa-e-Otimiza Orientado-a-Defeitos. Os experimentos computacionais demonstraram que a formulação alternativa é provê limites inferiores mais fortes que a original e que acelera o processo de busca por soluções inteiras. A abordagem de geração de colunas produziu limites inferiores ainda mais fortes. Resolvedores baseados em Fixa-e-Otimiza alcançaram resultados notáveis, sendo agora, por uma larga margem, os melhores resolvedores para programação de horários educacionais em XHSTT. A variação Fixa-e-Otimiza Orientado-a-Defeitos apresentou resultados promissores, superando sua versão original e auxiliando na busca de novas melhores soluções conhecidas. As metaheurísticas desenvolvidas também foram capazes de encontrar boas soluções para o problema. Por fim, os algoritmos e formulações propostos nessa tese resolveram seis instâncias em aberto e geraram quinze novas melhores soluções conhecidas de dezesseis instâncias e cinco novos limites inferiores.

Abstract

The Educational Timetabling Problem consists in assigning timeslots and resources to events respecting a set of constraints. Beyond its practical importance, this problem is classified as $\mathcal{NP}$-Hard in most of its formulations, attracting the interest of Computer Science and Operations Research scientific communities. This thesis considers the XHSTT format due to it is capability of covering several timetabling features and its relevance in recent literature. Moreover, this format was adopted in the last timetabling competition. The main goal of this work is to develop new formulations and algorithms to the problem. Regarding formulations, an alternative formulation and a Dantzig-Wolfe column generation approach were proposed. In the field of algorithms, two metaheuristics and some variants of them were developed. A problem-specific Fix-and-Optimize matheuristic approach was also proposed, along with a variation called Defect-Oriented Fix-and-Optimize. The computational experiments demonstrated that the proposed alternative formulation provides stronger lower bounds than the original one and speeds up the search for integer solutions. The column generation approach provided even stronger lower bounds. The Fix-and-Optimize based solvers achieved remarkable results being now, by a large margin, the best overall solvers for XHSTT educational timetabling. Defect-Oriented Fix-and-Optimize have shown promising results, overcoming its original version and helping to find new best known solutions. Metaheuristics and most of their variants were also able to find good solutions to the problem. Finally, the algorithms and formulations proposed in this thesis provided fifteen new best known solutions out of sixteen open instances and five new lower bounds. Six instances were closed in this work.

Acknowledgements

First of all, I would like to thank *God* for giving me strength and health to work on this P.hD. thesis.

I would like to thank my advisor, *Eduardo G. Carrano*, for his excellent guidance throughout this project, for his attention, and for his availability to support the development of this work. A special thanks goes to my co-advisor *Haroldo G. Santos*, for our productive meetings, for his amazing ideas, and also for his support to this project.

Great thanks to my department colleagues from DECSI/UFOP for their incentive and for letting me fully dedicate a year and a half to this thesis. Thanks goes also to my PPGEE/UFMG colleagues and staff for their support and for the good time spent together. Finally, thanks to the Brazilian research agency CNPq for the financial support to my exchange research stay in Denmark.

Thanks goes to all the fellows I met at DTU Management Engineering, specially to *Thomas R.S. Stidsen* for receiving me with open arms at DTU and for his insights to this work, and to *Niels-Christian F. Bagger* for his friendship and fruitful discussions. Really thank you for making my stay in Denmark simply awesome!

Finally, thanks goes to my family, friends, and to everyone else who supported me somehow in this project!

Contents

List of Figures

List of Tables

List of Acronyms

ALNS: Adaptive Large Neighborhood Search

BKS: Best Know Solution

CBT: Cluster Busy Times Cut

DO-FixOpt: Defect-Oriented Fix-and-Optimize

EBS: Event Block Swap

EM: Event Move

ES: Event Swap

FixOpt: Fix-and-Optimize

GD: Great Deluge

GUI: Graphical User Interface

ITC: International Timetabling Competition

IP: Integer Programming

KHE: Kingston High School Timetabling Engine

KM: Kempe Move

LAHC: Late Acceptance Hill-Climbing

LNS: Large Neighbourhood Search

LXY: Link X and Y Cut

LYQ: Link Y and Q Cut

MaxSAT: Maximum Boolean Satisfiability

MCF: Multi-commodity Flow

MIP: Mixed Integer Programming

MP: Mathematical Programming

NBT: Number of Busy Times Cut

RNA: Random Non-Ascendent

RPGD: Random Permutation Great Deluge

RVNS: Reduced Variable Neighborhood Search

RM: Resource Move

RS: Resource Swap

RPGD: Reduced Variable Neighborhood Search

SA: Simulated Annealing

SA-SF-LAHC: Simulated Annealing - Stagnation Free Late Acceptance Hill-Climbing

SAT: Boolean Satisfiability

SF-LAHC: Stagnation Free Late Acceptance Hill-Climbing

SVND: Sequential Variable Neighborhood Descent

SVNS: Skewed Variable Neighborhood Search

SVNS-DO-FixOpt: Skewed Variable Neighborhood Search - Defect-Oriented Fix-and-Optimize

SVNS-FixOpt: Skewed Variable Neighborhood Search - Fix-and-Optimize

VNS: Variable Neighborhood Search

XHSTT: eXtended Markup Language for High School Timetabling

Chapter 1

Introduction

Educational timetabling is a problem faced by many institutions around the world. It consists in assigning timeslots and resources to events taking into account several constraints. The schedule, which is generally made for a week, is repeated until the end of the class period. Under the umbrella of educational timetabling lie high school timetabling [60], university course timetabling [8], examination timetabling [64], and student sectioning [47]. Some usual constraints in these problems are to respect the availability of teachers, to respect the limit of lessons of the same class in a day, and to avoid idle times between activities.

The automated construction of educational schedules has been the subject of many research works in Computer Science and Operations Research. Surveys [68] and [60] present some reasons for this interest:

complexity: to find a timetable that satisfies the interests of all involved elements is a hard task; moreover, even the construction of a feasible timetable is a difficult problem;

practical importance: a good timetable can improve the students' performance, the staff satisfaction and it allows the institution to be more efficient in resource management;

theoretical importance: several formulations of the problem are classified as $\mathcal{NP}$-Hard [29].

This thesis considers the eXtended Markup Language for High School Timetabling (XHSTT) format [62] for modelling the problem. This format was chosen due its capacity of covering several timetabling features and due to its relevance in the recent literature, being adopted by the last international timetabling competition [61]. Although it was initially proposed for High School Timetabling, it can be also applied to universities and other types of educational institutions, such as shown in [33]. Indeed, several XHSTT problem sets found in the literature contain university timetabling, student sectioning, and other instances that are not necessarily from high school institutions. Finally, several instances are available in this format and the research community keeps track of the best known bounds for them.

1.1 Objectives

Despite the recent advances in the use of exact methods for timetabling, these techniques are still incipient for the XHSTT model. The model was proposed in 2010, but the first Integer Programming (IP) formulation was proposed only in 2014, by Kristiansen *et al.* [48]. In the field of heuristics, researchers believe that there is still room for improvement in their approaches [35]. With the proposal of the IP model, matheuristics [50] arise as candidate methods for solving the problem. With these techniques in mind, the main objective of this thesis is to develop, to improve, and to evaluate formulations and algorithms for educational timetabling. As specific objectives and contributions of this thesis, it should be mentioned:

- to evaluate the efficiency of metaheuristics applied to this problem;
- to build problem-specific matheuristics for educational timetabling;
- to propose alternative mathematical programming formulations for this problem;
- to develop a column generation decomposition for educational timetabling;

- to improve the best known solutions and lower bounds for the open instances of this problem in literature.

1.2 Literature Review

Computational study of educational timetabling has begun in 1963, by Gotlieb [37]. Since this initial work, a large number of approaches were applied to the problem. These approaches range from simple heuristics to complex techniques, such as metaheuristics, hyper-heuristics, and mixed integer programming formulations. Three international competitions on the theme were also organized.

Most early techniques were based on the simulation of human strategies for solving the problem. Such techniques are often refereed to as *direct heuristics* based on a *successive augmentation*. In this kind of algorithm, starting by the most difficult lecture, a timetable is iteratively built lecture by lecture, until all lectures are scheduled [69]. Reductions of the problem to the Graph Colouring Problem [55] and to Network Flow [58] were also explored. However, those techniques can only handle a few problem constraint types.

Nowadays, the most commonly techniques employed are metaheuristics. A metaheuristic is an algorithm that adopts any mechanism to escape from local optima in the optimization process. Any metaheuristic needs an initial solution as starting point, or a set of initial solutions, in the case of population-based methods. Usually, the initial solution is built through graph algorithms, direct heuristics, or by the combination of these techniques. Among the most used metaheuristics for timetabling it is possible to highlight: Simulated Annealing [1, 17], Greedy Randomized Adaptive Search [36, 53], Tabu Search [23, 73], Variable Neighborhood Search [16, 56], Iterated Local Search [6, 25], and Evolutionary Strategies [15, 28].

Although metaheuristics frequently achieve reasonable solutions in short time, they do not ensure optimality. Moreover, these methods cannot even provide optimality

gaps. In this sense, exact techniques, such as Constraint Programming and Mixed Integer Programming, take an important role. They can provide lower bounds to evaluate more precisely the quality of a solution and even reach good or optimal solutions. However, to achieve good/ optimal solutions using exact methods is possible only when the timetabling problem is too small or when advanced techniques of formulation are applied, such as Column Generation [7], Cuts [40], and Decomposition [19]. For example, (*i*) Achá and Nieuwenhuis [4] improved several best known solutions for instances from the Second International Timetabling Competition using a boolean satisfiabily formulation (SAT) in a framework that maximizes the amount of clauses satisfied (MaxSAT); (*ii*) Santos *et al.* [67] presented a column and cut generation algorithm based on an Integer Programming formulation that was able to find optimal solutions for some Brazilian school timetabling instances, and; (*iii*) Avella and Vasilev [5] studied the polyhedral structure of the University Course Timetabling problem to provide effective classes of cutting planes. As result they found, within fifteen minutes, the optimal solution to the four instances considered in their work, whereas the compact formulation could not find any feasible solution within two hours.

Although educational timetabling problems share some core features, they present several particularities from country to country. Some examples are given below:

Australia : Australian educational institutions have short scholar days and high utilization of teachers in these days. Therefore it is highly desirable a compact schedule for teachers. Specialized rooms, such as science and computing laboratories are highly demanded. Constraints related to the teachers' workload are also critical [1, 42].

Brazil : lectures are weekly distributed in Brazilian schools and universities. Teachers often work in more than one institution, therefore constraints related to the availability of them are crucial. Constraints related to the duration of lectures are important as well [66, 73].

England : English schools have high variation among the duration of lectures. They might be cycled or distributed over two weeks. Additionally, students can enroll in elective subjects of their preference. Constraints related to the duration and splitting of events are important in their instances [78].

Finland : in Finn educational institutions, students attend to lectures from their base group and should also attend some elective lectures. Usually teachers and rooms are pre-assigned according to the compatibility with the events. Compact schedules to the students are mandatory and idle times between lectures are highly undesirable [57].

Greece Greek high schools are divided in two parts: *lyceum* (7th to 9th grades) and *gymnasium* (10th to 12th grades). Students must choose between three distinct specializations, each one with a different set of lectures. Lectures are distributed in five days, having seven times each. This schedule is weekly repeated. Constraints related to the duration of the lectures are critical. Idle times between teacher's lectures are undesirable [9, 75].

Netherlands : in Dutch schools, the schedule is weekly repeated and it may last six weeks, one trimester, one semester, or the whole year. Usually teachers are pre-assigned to the lectures and work contracts allow one or two days off. Therefore, avoiding unavailable times for teachers is essential. Students gifted for sports or dance are released from the first or the last lecture on some days [22, 77].

Due to this variety of timetabling models and the difficulty of establishing fair comparisons among solution approaches, three International Timetabling Competitions (ITC) were organized:

ITC2002 : the first competition occurred in 2002/03 and it was focused on the University Course Timetabling Problem. The best ranked solver in the first ITC employed a 3-phase approach: first, a feasible timetable is constructed using graph

colouring and maximum matching; then, Simulated Annealing (SA) is used to order the timeslots built, and; finally, SA is used to swap individual events between timeslots to improve the solution quality [46].

ITC2007 : the second competition occurred in 2007/08 and it was divided into three tracks (Examination Timetabling [51], Post Enrolment based Course Timetabling [49] and Curriculum based Course Timetabling [25]). The overall best ranked solver was also a multiphase method. In the construction phase, a complete solution is found using an Iterative Forward Search algorithm. In the next phase a local optimum is found using a Hill Climbing algorithm. Afterwards, a modified version of Great Deluge (GD) technique is used. Simulated Annealing is sometimes applied between GD iterations [54].

ITC2011 : the third competition occurred in 2011/12 and it was the first to adopt the XHSTT format. Nowadays, more than 40 real world datasets, from 12 different countries, are available in such a format [61]. They can be used to evaluate the performance of algorithms for high school/university timetabling. The competition was divided into three phases: the first one was dedicated to improve best known solutions from public instances; the second one aimed to evaluate the solver performances over hidden instances in a constrained time, and; the third one focused on improving best known solutions from the hidden instance set. The best overall ranked solver was GOAL, followed by, respectively, Lectio, HySTT and HFT. More detail of these solvers and recent work on XHSTT format is given in sequence.

Fonseca *et al.* [35] developed a hybrid algorithm to solve the XHSTT timetabling problem. They participated of ITC2011 under the codename GOAL. In the approach, an initial solution is generated using Kingston High School Timetabling Engine (KHE). This solution is improved through Simulated Annealing and then Iterated Local Search, using seven neighbourhood moves.

Sørensen *et al.* [72] participated in the ITC2011 under the codename Lectio and developed an Adaptive Large Neighbourhood Search (ALNS) approach for XHSTT. Their algorithm was composed of three ALNS strategies: remove strategy, adaptive strategy, and accept strategy. Remove strategy makes several unassignments following problem-related rules. The unassignments might be of times, resources, or both. Adaptive strategy makes several greedy assignments at a time (for instance, testing all possible assignments for each event at a time). The accept strategy is a temperature based algorithm, similar to Simulated Annealing.

Khieri *et al.* [41] developed a hyper-heuritstic for XHSTT and participated in the ITC2011 as HySTT. They used the KHE library to generate initial solutions and, afterwards, applied a hyper-heuristic algorithm. The developed hyper-heuristic consists of a framework that controls 11 low-level heuristics. The low-level heuristics are divided in two sets: mutational and hill-climbing. Mutational operators do randomized moves that are accepted when they do not degrade the incumbent solution more than an acceptance factor ϵ. Afterwards, hill-climbing heuristics try to improve the mutated solution. Recently they released an improved version of the hyper-heuristic solver, where the best performing solver is Random Permutation Great Deluge (RPGD) [3].

Romros and Homberger [26] developed an Evolutionary Algorithm that was labelled as HFT in ITC2011. In their approach, initially a solution is randomly generated considering the requirements to split events into sub-events. Afterwards, times and resources are assigned to the sub-events through an insertion heuristic. While the computational time has not been reached, a mutation step generates a new split of events based on the incumbent split and the insertion heuristic is called again to assign times and resources. The insertion heuristic searches, for each sub-event, a time-resources assignment that does not violate any hard constraint. In such a case, the assignment is done; otherwise, the sub-event is left unassigned.

More recently, Demirović and Musliu [24] proposed a maximum satisfiability (MaxSAT) based approach for XHSTT timetabling. In their approach a local search algo-

rithm is used to drive an initial solution into a local optimum and then more powerful large neighbourhood search (LNS) techniques based on Max-SAT are used to further improve the solution. The proposed Max-SAT LNS has a destroy operator that removes the assignments of all events related to a pair of resources or removes all assignments of a given pair of days in the week. The insert operator tries to find the best insertion for the unassigned events using an exhaustive search based on a Max-SAT formulation. One limitation of the proposed Max-SAT approach is that the assignment of resources could not be modelled, therefore it cannot be applied to instances that have this feature.

Although several solution approaches have been proposed for this problem, the XHSTT problem model is relatively recent and, consequently, only a few of them tackle this specific format. Moreover it is believed that there is still room for improvement in the algorithmic approaches to this format. For instance, no problem specific matheuristic has yet been proposed for this model. In the field of mathematical programming approaches no much effort has been put to develop alternative formulations to the existing IP model. Therefore, this is the gap that this thesis aims to cover in the literature of automated educational timetabling.

1.3 Document Structure

This thesis is organized into six chapters. In Chapter 2, the XHSTT format is explained in detail along with an integer programming formulation to handle it. Chapter 3 describes the proposed integer programming techniques for this timetabling problem model, which includes an alternative formulation that applies cuts and preprocessing techniques, a column generation decomposition, and a cut-and-solve approach. Chapter 4 presents the algorithms developed for this problem. It contains the method to generate initial solutions, the neighbourhood structure, metaheuristics developed, adaptations proposed, and a problem-specific matheuristic for timetabling. Chapter 5 presents the computational environment, the instance sets considered, the obtained results, dis-

cussion, and the new best known solutions and lower bounds achieved in this work. Finally, Chapter 6 presents the concluding remarks, the contributions of this thesis, and suggestions of future work on automated educational timetabling.

Chapter 2

The Timetabling Problem

This thesis considered the XHSTT model to represent the educational timetabling problems. The XHSTT format was proposed by Post *et al.* [62] with the purpose of becoming a standard format that covers the most important features of educational timetabling. Such a model is split into four main entities: (*i*) Times, (*ii*) Resources, (*iii*) Events, and (*iv*) Constraints. A solution consists of a set of assignments of times and resources to the events. Each entity is explained in detail in the next sections, as well as a complete integer programming formulation for this problem.

2.1 Times

Entity *Times* contains information related to the timeslots available for assignments. A set of timeslots can also be grouped into a *Time Group*. It is common to group together the times of a given week day and/or the morning/afternoon times into a Time Group.

2.2 Resources

Entity *Resources* contains information about the resources that the educational institution has. Each resource is of a specific *Resource Type*. Usual resource types are: class, teacher, room, and laboratory. However, this is an abstract concept in the

format and any type of resource can be specified. Resources can also be grouped into so called *Resource Groups*. A resource group can specify, for example, a set of teachers of mathematics, a set of night classes, and a set of rooms having some specific feature.

2.3 Events

Entity *Events* contains information related to the events that have to be scheduled. An *Event* is a meeting between resources. It specifies that the resources should meet for a given number of times. Events can also be grouped into *Event Groups* when they share some specific feature, for example Gymnastic or Science lectures. An XHSTT event has:

Duration: the number of times in which it has to be assigned.

Workload (optional): a number that will be added to the workload of the resources assigned to attend it.

Time (optional): the preassigned time for it. It is often absent, if so, the XHSTT solver should assign a set of times for the event.

Resources: the set of resources that attends the event. When a resource is preassigned, its identifier is given; otherwise, a resource of the proper type should be assigned by the solver. Each resource plays a specific role in the event. The role is later used to link it to certain constraints.

2.4 Constraints

Constraints define what is required and what is desired in a solution for a timetabling problem. The constraints are divided into hard constraints, whose compliance is mandatory; and soft constraints, whose satisfaction is only desirable. Each constraint can be enabled or disabled in each instance and, when enabled, specified as either hard or soft.

Every constraint violation implies a deviation, which must be combined with the constraint weight and with the cost function type to define how the constraint is penalized in the objective function. There are three types of cost functions: Linear, Quadratic, and Step. For constraints whose cost function is linear, the non-compliance penalty is equal to the number of deviations multiplied by the weight of the constraint. For quadratic cost constraints, the square of the number of deviations, multiplied by the weight, will be added to the objective function. Finally, for step type constraints, the weight will be added only once to the objective function, independently on how many deviations occurred.

There are 16 constraint types in the XHSTT format:

1. **Assign Time**: specifies the necessity of assigning enough timeslots for a set of events;
2. **Assign Resource**: specifies the requirement of assigning resources for a set of events;
3. **Prefer Times**: indicates that some event has preference for a particular set of timeslots;
4. **Prefer Resources**: indicates that some event has preference for a particular set of resources.
5. **Link Events**: specifies that a set of events should occur at the same time;
6. **Order Events**: enforces that one event should occur at least/most a certain number of timeslots after another;
7. **Spread Events**: indicates that a set of events should be spread in at least/most a certain number of days;
8. **Avoid Split Assignments**: specifies that the same resource should be assigned for all meets of an event;

9. **Distribute Split Events**: indicates that a set of events has to be split between a minimum and a maximum number of meets of a given duration;

10. **Split Events**: defines limits on the number of non-consecutive meets created for an event and on their durations.

11. **Avoid Clashes**: states that resources must be assigned without clashes (i.e. without assigning the same resource to more than one event per timeslot);

12. **Avoid Unavailable Times**: specifies that certain resources are unavailable to attend any event at some timeslots;

13. **Limit Workload**: restricts the workload of a given resource between minimum and maximum bounds;

14. **Limit Idle Times**: states that the number of idle times per day must lie between a minimum and a maximum bound for a set of resources. An idle time is a time with no event assignment between other times with assignments within the same day;

15. **Limit Busy Times**: indicates that the number of busy times in a day should lie between a minimum and a maximum bound for a set of resources;

16. **Cluster Busy Times**: specifies bounds on the number of days in which a resource can be attending activities.

The objective function $f(.)$ is calculated in terms of constraint violations. Each violation is penalized according to its weight (minimization problem). The function value is in fact a pair (H, S): feasibility value (H), which counts the violation of hard constraints, and quality value, which counts the violation of soft constraints (S). For example, (2, 51) represents a cost of two unities of feasibility and 51 unities of quality. The solutions are first compared in terms of H and if two solutions have the same

performance for feasibility, then they are compared in terms of S. When the feasibility violation cost is 0 (feasible solution), H is usually omitted from the notation.

2.5 Integer Programming Formulation

A complete Integer Programming formulation for XHSTT was proposed by Kristiansen *et al.* [48]. This formulation was designed to describe precisely the XHSTT format. However, before this thesis, little effort has been made to improve it. This formulation will be presented in this section and denoted as $\mathscr{F}_1$ throughout the thesis.

The formulation takes as main input sets:

$\mathcal{T}$	Times
$\mathcal{TG}$	Time Groups
$\mathcal{R}$	Resources
$\mathcal{RG}$	Resource Groups
$\mathcal{E}$	Events
$\mathcal{EG}$	Event Groups
$\mathcal{C}$	Constraints

An event $e \in \mathcal{E}$ has a duration $D_e \in \mathbb{N}$ and a demand for a set of resources (event resources), denoted as $er \in \mathcal{ER}_e$. Furthermore, a resource demanded for the event er can undertake a role $role_{er}$, which is used to link the resource to certain constraints. A resource $r \in \mathcal{R}$ can be preassigned to fulfil the demand $er \in \mathcal{ER}_e$. Parameter $\rho_{er} \in \{1, 0\}$ takes value 1 if event resource er has a preassigned resource, and 0 otherwise, while parameter $\hat{\rho}_{er,r} \in \{1, 0\}$ takes value 1 if resource r is preassigned to event resource er, and 0 otherwise. A sub-event se is defined as a fragment of a specific event $e \in \mathcal{E}$. Each sub-event has a duration $D_{se} \leq D_e$ and inherits exactly the same resource requirements of the source event.

Let $\mathcal{SE}$ be the entire set of sub-events of a XHSTT instance and let $se \in \mathcal{SE}_e$ be a set of the sub-events generated from an event e. The total duration of all sub-events generated from event e in a solution must be exactly D_e. A set of all possible sub-events with different durations is created, such that all combinations of sub-events for a given

event can be handled. For example, if an event has duration 4, the set of sub-events for this event has the respective lengths: 1, 1, 1, 1, 2, 2, 3, and 4. Thereby the set of possible sub-events for an event $e \in \mathcal{E}$ with duration D_e has $\sum_{i=1}^{D_e} \lfloor \frac{D_e}{i} \rfloor$ elements.

The set of resources and times are both extended with dummy-indices, denoted dummy-resource r_D and dummy-time t_D, respectively. They are necessary to handle the unusual case of an optimal solution in which one or more events do not have resources or start times assigned.

Times $\mathcal{T}$ are organized in chronological order. Thus, p_t denotes the index number of time $t \in \mathcal{T}$. A time group $tg \in \mathcal{TG}$ defines a set of times, in such a way $t \in \mathcal{T}_{tg}$ denotes the times belonging to time group tg. Additionally

$$\mathcal{T}_{se,t}^{start} = \{t' \in \mathcal{T} \setminus \{t_D\} \mid p_t - D_{se} + 1 \leq p_{t'} \leq p_t\} \quad \begin{smallmatrix} \forall se \in \mathcal{SE} \\ \forall t \in \mathcal{T} \end{smallmatrix} \tag{2.1}$$

is pre-processed to denote the set of times that a sub-event $se \in \mathcal{SE}$ occurs assuming that it is assigned start time $t \in \mathcal{T}$.

Each constraint $c \in \mathcal{C}$ is of a specific type and it applies to certain events, resources, or event groups. Notations $e \in \mathcal{E}_c$, $r \in \mathcal{R}_c$, and $eg \in \mathcal{EG}_c$ represent, respectively, the events, resources, or event groups that a constraint applies to.

2.5.1 Variables

The main decision variables of this formulation are $x_{se,t,er,r}$ binary variables:

$$x_{se,t,er,r} = \begin{cases} 1 & \text{if sub-event } se \text{ is assigned to start time } t \text{ and resource} \\ & r \text{ is assigned to event resource } er. \\ 0 & \text{otherwise.} \end{cases}$$

additionally, the following auxiliary variables are used:

$y_{se,t}$ = 1 if sub-event se is assigned to start time t; 0 otherwise.
$w_{se,er,r}$ = 1 if sub-event se is assigned to resource r for event resource er; 0 otherwise.
u_{se} = 1 if sub-event se is active; 0 otherwise.
$v_{t,r}$ = number of times in which resource r is used at time t.
$q_{r,t}$ = 1 if resource r is busy at time t; 0 otherwise.
$p_{r,tg}$ = 1 if resource r is busy at at least one time of time group tg; 0 otherwise.
$o_{e,t}$ = 1 if at least one sub-event of event e is assigned to time t; 0 otherwise.
$l_{eg,t}$ = 1 if at least one event of event group eg is assigned to time t; 0 otherwise.
$k_{eg,r}$ = 1 if resource r is assigned to at least one event in event group eg; 0 otherwise.
$h_{r,tg,t}$ = 1 if resource r has an idle time in time t in time group tg; 0 otherwise.
h_e^{first} = ordinal number of the first time assigned to any sub-event of event e.
h_e^{last} = ordinal number of the latest time assigned to any sub-event of event e.

Each constraint $c \in \mathcal{C}$ has a set of points-of-application which, in turn, might be any XHSTT entity depending on the constraint that it is related to. To simplify the notation, in some equations points-of-application will be denoted as $p \in \mathcal{P}_c$ regardless of the entity. Each point-of-application is associated with a set of deviations, indexed by $d \in \mathcal{D}_p$, and each set of deviations has an associated non-negative cost. Therefore the slack variables

$$s_{c,p,d} = \text{value of deviation } d \text{ of point-of-application } p \text{ in constraint } c.$$

are used to calculate the penalties for each XHSTT constraint.

2.5.2 Constraints

In addition to all constraints described in the XHSTT specification, some basic constraints are required to ensure the consistency of the model. First of all, it is necessary to make sure that only one start time is assigned to a sub-event, and the number of resources assigned is exactly the same number of the event resources required by the event:

$$\sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}_{er}} x_{se,t,er,r} = 1 \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall er \in \mathcal{ER}_{se} \end{matrix} \tag{2.2}$$

Constraint set (2.3) makes the link between variables $x_{se,t,er,r}$ and $y_{se,t}$, in which $|\mathcal{ER}_{se}|$ denotes the number of event resources demanded for a sub-event se.

$$\sum_{er \in \mathcal{ER}_{se}} \sum_{r \in \mathcal{R}_{er}} x_{se,t,er,r} = |\mathcal{ER}_{se}| \times y_{se,t} \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall t \in \mathcal{T} \end{matrix} \tag{2.3}$$

The link from variables $x_{se,t,er,r}$ to variables $v_{t,r}$ is shown in constraint set (2.4). For each time $t \in \mathcal{T} \setminus \{t_D\}$ and resource $r \in \mathcal{R}$, $v_{r,t}$ will be active iff $x_{se,t',er,r}$ is active. Recall that t' represents the times that will be used if se is assigned t as starting time.

$$\sum_{se \in \mathcal{SE}} \sum_{er \in \mathcal{ER}_{se}} \sum_{t' \in \mathcal{T}^{start}_{se,t}} x_{se,t',er,r} = v_{t,r} \quad \begin{matrix} \forall t \in \mathcal{T} \setminus \{t_D\} \\ \forall r \in \mathcal{R} \end{matrix} \tag{2.4}$$

The link between variables $x_{se,t,er,r}$ and $w_{se,er,r}$ is done as follows:

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall er \in \mathcal{ER}_{se} \setminus \{t_D\} \\ \forall r \in \mathcal{R} \end{matrix} \tag{2.5}$$

Constraint set (2.6) ensures that a starting time $t \in \mathcal{T}$ is not assigned to a sub-event $se \in \mathcal{SE}$ if there is not enough continuous times after t to accommodate the duration of se:

$$y_{se,t} = 0 \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall t \in \mathcal{T} \setminus \{t_D\} \ : \ p_t + D_{se} - 1 > |\mathcal{T}| \end{matrix} \tag{2.6}$$

Although all possible sub-events for an event are created, only a subset of them should be active in the final solution. Recall that a sub-event is active if a starting time or a resource is assigned to it. Constraint sets (2.7), (2.8), and (2.9) are imposed to ensure the correct activation of the sub-events.

$$\sum_{r \in er \setminus \{r_D\}} w_{se,er,r} \leq u_{se} \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall er \in \mathcal{ER}_{se} \ : \ \rho_{er} = 0 \end{matrix} \tag{2.7}$$

$$\sum_{t \in \mathcal{T} \setminus \{t_D\}} y_{se,t} \leq u_{se} \quad \forall se \in \mathcal{SE} \tag{2.8}$$

$$\sum_{t \in \mathcal{T} \setminus \{t_D\}} y_{se,t} + \sum_{er \in \mathcal{ER}_{se}: \rho_{er}=0} \sum_{r \in \mathcal{R}_{er} \setminus \{r_D\}} w_{se,er,r} \geq u_{se} \quad \forall se \in \mathcal{SE} \tag{2.9}$$

The sum of the durations of the sub-events of a given event must be equal to the duration of the source event:

$$\sum_{se \in \mathcal{SE}_e} u_{se} \times D_{se} = D_e \quad \forall e \in \mathcal{E} \tag{2.10}$$

A resource is busy at some time if it attends to at least one event at that time, and it is busy at some time group if it is busy at one or more times within the times of that time group. The values of variables $q_{r,t}$ and $p_{r,tg}$ are set by the following constraints:

$$|\mathcal{SE}| \times q_{r,t} \geq v_{t,r} \quad \begin{smallmatrix} \forall r \in \mathcal{R} \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{smallmatrix} \tag{2.11}$$

$$q_{r,t} \leq v_{t,r} \quad \begin{smallmatrix} \forall r \in \mathcal{R} \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{smallmatrix} \tag{2.12}$$

$$p_{r,tg} \geq q_{r,t} \quad \begin{smallmatrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \\ \forall t \in \mathcal{T}_{tg} \end{smallmatrix} \tag{2.13}$$

$$p_{r,tg} \leq \sum_{t \in \mathcal{T}_{tg}} q_{r,t} \quad \begin{smallmatrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \end{smallmatrix} \tag{2.14}$$

Constraints (2.11) and (2.13) establish lower bounds for variables $q_{r,t}$ and $p_{r,tg}$. They ensure that these variables must take value 1 if the resource is busy at the respective time/time group. Constraints (2.12) and (2.14) are necessary to ensure that variables $q_{r,t}$ and $p_{r,tg}$ are 0 if the resource is not busy at the respective time/time group.

In sequence, each XHSTT specific constraint type is formulated. Let set $\mathcal{C}^i \subseteq \mathcal{C}$ denote all constraints of a certain type, as follows:

$\mathcal{C}^1$ – **Assign Resource** : An assign resource constraint penalizes a solution when no resource is assigned to supply a demand of an event resource. Specifically, the deviation at one point-of-application is the sum of the durations of the sub-events of the respective event in which a resource is not assigned. The deviation $s^1_{c,er}$ at each point-of-application of this constraint is calculated through the following equation:

$$D_e - \sum_{se \in \mathcal{SE}_e} \sum_{r \in \mathcal{R}_{er} \setminus \{r_D\}} D_{se} \times w_{se,er,r} = s^1_{c,er} \quad \begin{matrix} \forall c \in \mathcal{C}^1 \\ \forall e \in \mathcal{E}_c \\ \forall er \in \mathcal{ER}_e \,:\, role_{er} = role_c \end{matrix} \tag{2.15}$$

$\mathcal{C}^2$ – **Assign Time** : The assign time constraint penalizes sub-events in which times are not assigned. The deviation $s^2_{c,e}$ at one point-of-application is the total duration of those sub-events derived from the specific event in which a time is not assigned.

$$D_e - \sum_{se \in \mathcal{SE}_e} \sum_{t \in \mathcal{T} \setminus \{t_D\}} D_{se} \times y_{se,t} = s^2_{c,e} \quad \begin{matrix} \forall c \in \mathcal{C}^2 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.16}$$

$\mathcal{C}^3$ – **Split Events** : A split events constraint defines limits to the number of sub-events that can be derived from a given event and to their durations. Let the parameters $\underline{B}^{amt}_c \in \mathbb{N}$ and $\overline{B}^{amt}_c \in \mathbb{N}$ be, respectively, the minimum and the maximum number of sub-events in which a given event can be split, and $\underline{B}^{dur}_c \in \mathbb{N}$ and $\overline{B}^{dur}_c$ be the minimum and maximum durations of such sub-events.

The value of the deviation at each point-of-application (each event) of this constraint is given by the number of sub-events of the source event whose duration is lower than $\underline{B}^{dur}_c$ or greater than $\overline{B}^{dur}_c$ ($s^{3a}_{c,e}$) and the number of sub-events below

$\underline{B}_c^{amt}$, or above $\overline{B}_c^{amt}$ ($s_{c,e}^{3b}$). The following constraint sets are imposed:

$$\sum_{\substack{se \in \mathcal{SE}_e : \\ \underline{B}_c^{dur} > D_{se} \vee \overline{B}_c^{dur} < D_{se}}} u_{se} = s_{c,e}^{3a} \quad \begin{matrix} \forall c \in \mathcal{C}^3 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.17}$$

$$\underline{B}_c^{amt} - \sum_{se \in \mathcal{SE}_e} u_{se} \leq s_{c,e}^{3b} \quad \begin{matrix} \forall c \in \mathcal{C}^3 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.18}$$

$$\sum_{se \in \mathcal{SE}_e} u_{se} - \overline{B}_c^{amt} \leq s_{c,e}^{3b} \quad \begin{matrix} \forall c \in \mathcal{C}^3 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.19}$$

The full deviation for constraint $c \in \mathcal{C}^3$ is given by $s_{c,e}^{3a} + s_{c,e}^{3b}$.

$\mathcal{C}^4$ – **Distribute Split Events** : Distribute split event constraints impose limits on the number of sub-events of a particular duration that may be derived from an event. Let $D_c \in \mathbb{N}$ be the duration of the sub-events for which this constraint applies, and let $\underline{B}_c$ and $\overline{B}_c$ be the minimum and maximum number of sub-events of duration D_c that may be derived from a given event, respectively.

$$\underline{B}_c - \sum_{\substack{se \in \mathcal{SE}_e \\ D_{se} = D_c}} u_{se} \leq s_{c,e}^{4} \quad \begin{matrix} \forall c \in \mathcal{C}^4 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.20}$$

$$\sum_{\substack{se \in \mathcal{SE}_e \\ D_{se} = D_c}} u_{se} - \overline{B}_c \leq s_{c,e}^{4} \quad \begin{matrix} \forall c \in \mathcal{C}^4 \\ \forall e \in \mathcal{E}_c \end{matrix} \tag{2.21}$$

$\mathcal{C}^5$ – **Prefer Resources** : This constraint defines that an event resource has a preference for certain resources. The assignment of all non-preferred resources is taken and the duration of the sub-events in which these non-preferred resources are assigned is summed to calculate the deviation. Let $r \in \mathcal{R}_c$ denote a preferred

resource:

$$\sum_{se \in \mathcal{SE}_e} \sum_{r \in \mathcal{R} \setminus \{r_D\} \,:\, r \notin \mathcal{R}_c} D_{se} \times w_{se,er,r} = s^5_{c,er} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^5 \\ \forall e \in \mathcal{E}_c \\ \forall er \in \mathcal{ER}_e \,:\, \rho_{er}=0 \wedge role_{er}=role_c \end{smallmatrix} \tag{2.22}$$

$\mathcal{C}^6$ – **Prefer Times** : Like the prefer resources constraint, events might also have preferences for certain times. The deviation is calculated for each event by summing the duration of all sub-events which are assigned a time that is not in the preferred times list. The constraint has an optional duration-property, denoted $D_c \in \mathbb{N}_0$. If this property is given, only sub-events of duration D_c are considered. Let $t \in \mathcal{T}_c$ denote a preferred time:

$$\sum_{se \in \mathcal{SE}_e} \sum_{t \in \mathcal{T} \setminus \{t_D\} : t \notin \mathcal{T}_c \wedge D_c = D_{se}} D_{se} \times y_{se,t} = s^6_{c,er} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^6 \\ \forall e \in \mathcal{E}_c \end{smallmatrix} \tag{2.23}$$

$\mathcal{C}^7$ – **Avoid Split Assignments** : When an event is split into sub-events, each of its event resources is also split for each sub-event. A different resource may be assigned to each of these generated event resources. This constraint penalizes the assignment of different resources to event resources within the same event group. The constraint examines the demand of all event resources derived from the events in the event group and it calculates the number of distinct resources assigned to them, ignoring unassigned event resources. The deviation $s^7_{c,eg}$ is the number of resources that exceeds 1.

$$\sum_{\substack{er \in \mathcal{ER}_e, \rho_{er}=0 \\ role_c = role_{er}}} w_{se,er,r} \leq k_{eg,r} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^7 \\ \forall r \in \mathcal{R} \\ \forall eg \in \mathcal{EG}_c \\ \forall e \in \mathcal{E}_{eg} \\ \forall se \in \mathcal{SE}_e \end{smallmatrix} \tag{2.24}$$

$$\sum_{r \in \mathcal{R}} k_{eg,r} - 1 \leq s^7_{c,eg} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^7 \\ \forall eg \in \mathcal{EG}_c \end{smallmatrix} \tag{2.25}$$

$\mathcal{C}^8$ – **Spread Events** : The spread events constraint has a deviation for each time group $tg \in \mathcal{TG}_c \in \mathcal{C}^8$. Let $\underline{B}_{c,tg}$ and $\overline{B}_{c,tg}$ be, respectively, the minimum and maximum number of sub-events of a given event that can be placed in time group tg of constraint c. The deviation $s^8_{c,eg,tg}$, for each time group, is given by the number of assignments of sub-events from event group eg that is below $\underline{B}_{c,tg} \in \mathbb{N}$ or above $\overline{B}_{c,tg} \in \mathbb{N}$.

$$\underline{B}_{c,tg} - \sum_{se \in \mathcal{SE}_e} \sum_{e \in \mathcal{E}_{eg}} \sum_{t \in \mathcal{T}_{tg}} y_{se,t} \leq s^8_{c,eg,tg} \quad \begin{matrix} \forall c \in \mathcal{C}^8 \\ \forall eg \in \mathcal{EG}_c \\ \forall tg \in \mathcal{TG}_c \end{matrix} \tag{2.26}$$

$$\sum_{se \in \mathcal{SE}_e} \sum_{e \in \mathcal{E}_{eg}} \sum_{t \in \mathcal{T}_{tg}} y_{se,t} - \overline{B}_{c,tg} \leq s^8_{c,eg,tg} \quad \begin{matrix} \forall c \in \mathcal{C}^8 \\ \forall eg \in \mathcal{EG}_c \\ \forall tg \in \mathcal{TG}_c \end{matrix} \tag{2.27}$$

$\mathcal{C}^9$ – **Link Events** : A link event constraint specifies that a set of events within an event group should be assigned to the same starting time. The deviation of this constraint is set as the number of times in which at least one event in the event group does not occur simultaneously with the others. Constraints (2.28), (2.29) and (2.30) ensure that variables $o_{e,t}$ and $l_{eg,t}$ assume correct values. The slack variable of Link Events constraint, $s^9_{c,eg,t}$, is defined in constraint (2.31).

$$\sum_{t' \in \mathcal{T}^{start}_{se,t}} y_{se,t'} \leq o_{e,t} \quad \begin{matrix} \forall e \in \mathcal{E} \\ \forall se \in \mathcal{SE}_e \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.28}$$

$$\sum_{se \in \mathcal{SE}_e} \sum_{t' \in \mathcal{T}^{start}_{se,t}} y_{se,t'} \geq o_{e,t} \quad \begin{matrix} \forall e \in \mathcal{E} \\ \forall se \in \mathcal{SE}_e \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.29}$$

$$l_{eg,t} \geq o_{e,t} \quad \begin{matrix} \forall eg \in \mathcal{EG} \\ \forall e \in \mathcal{E}_{eg} \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.30}$$

$$l_{eg,t} - o_{e,t} \leq s^{9}_{c,eg,t} \quad \begin{matrix} \forall c \in \mathcal{C}^{9} \\ \forall eg \in \mathcal{EG}_c \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.31}$$

$\mathcal{C}^{10}$ – **Order Events** : An order events constraint specifies that the times assigned to two events should be in order, in such a way that the first event ends before the second event starts. Let parameters $\underline{B}_c \in \mathbb{N}$ and $\overline{B}_c \in \mathbb{N}$ be, respectively, the minimum and maximum number of times that may separate two events. Let $(e, \hat{e}) \in (\mathcal{E}, \mathcal{E})_c$ denote an event pair such that this constraint applies to. The deviation, $s^{10}_{c,e,\hat{e}}$, is then given by the amount by which the difference between these h^{last}_e and h^{first}_e exceeds $\overline{B}_c$ or falls below $\underline{B}_c$.

$$p_t \times y_{se,t} + D_{se} \leq h^{last}_e \quad \begin{matrix} \forall c \in \mathcal{C}^{10} \\ \forall e \in \mathcal{E}_c \\ \forall se \in \mathcal{SE}_e \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.32}$$

$$|\mathcal{T}| - (|\mathcal{T}| - p_t) \times y_{se,t} \geq h^{first}_{\hat{e}} \quad \begin{matrix} \forall c \in \mathcal{C}^{10} \\ \forall \hat{e} \in \mathcal{E}_c \\ \forall se \in \mathcal{SE}_e \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.33}$$

$$\underline{B}_c - (h^{first}_{\hat{e}} - h^{last}_e) \leq s^{10}_{c,e,\hat{e}} \quad \begin{matrix} \forall c \in \mathcal{C}^{10} \\ \forall (e,\hat{e}) \in (\mathcal{E},\mathcal{E})_c \end{matrix} \tag{2.34}$$

$$(h^{first}_{\hat{e}} - h^{last}_e) - \overline{B}_c \leq s^{10}_{c,e,\hat{e}} \quad \begin{matrix} \forall c \in \mathcal{C}^{10} \\ \forall (e,\hat{e}) \in (\mathcal{E},\mathcal{E})_c \end{matrix} \tag{2.35}$$

$\mathcal{C}^{11}$ – **Avoid Clashes** : These constraints specify that certain resources should not have clashes in their timetables. It means they should not be assigned to two or more events simultaneously. This constraint produces a set of deviations for each resource. For each time, the number of occurrences of a given resource minus one is calculated to estimate the deviation of that resource for that time.

$$v_{t,r} - 1 \leq s^{11}_{c,r,t} \quad \begin{matrix} \forall c \in \mathcal{C}^{11} \\ \forall r \in \mathcal{R}_c \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{2.36}$$

$\mathcal{C}^{12}$ – **Avoid Unavailable Times** : An avoid unavailable times constraint specifies that certain resources are unavailable for any event at certain times. The deviation, $s^{12}_{c,r}$ is the number of times that are being attended by an unavailable resource. Let $t \in \mathcal{T}_c$ denote that t is an unavailable time for constraint $c \in \mathcal{C}^{12}$:

$$\sum_{t \in \mathcal{T}_c} q_{r,t} = s^{12}_{c,r} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^{12} \\ \forall r \in \mathcal{R}_c \end{smallmatrix} \tag{2.37}$$

$\mathcal{C}^{13}$ – **Limit Idle Times** : A resource is idle at some time $t \in \mathcal{T}_{tg}$ if it is not attending any sub-event at t, but it is attending events before and after t in the same time group tg. Limit idle times constraint limits the number of idle times a resource may have within a time group.

$$q_{r,\hat{t}} - q_{r,t} + q_{r,\hat{\hat{t}}} - 1 \leq h_{r,tg,t} \quad \begin{smallmatrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \\ \forall t,\hat{t},\hat{\hat{t}} \in \mathcal{T}_{tg} \,:\, p_{\hat{t}} < p_t < p_{\hat{\hat{t}}} \end{smallmatrix} \tag{2.38}$$

For each resource of the constraint, the deviation is the number of cases in which the amount of idle times is under $\underline{B}_c \in \mathbb{N}$ or above $\overline{B}_c \in \mathbb{N}$. Slack variable $s^{13}_{c,r}$ computes the sum of such cases.

$$\underline{B}_{c,r} - \sum_{tg \in \mathcal{TG}_c} h_{r,tg} \leq s^{13}_{c,r} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^{13} \\ \forall r \in \mathcal{R}_c \end{smallmatrix} \tag{2.39}$$

$$\sum_{tg \in \mathcal{TG}_c} h_{r,tg} - \overline{B}_{c,r} \leq s^{13}_{c,r} \quad \begin{smallmatrix} \forall c \in \mathcal{C}^{13} \\ \forall r \in \mathcal{R}_c \end{smallmatrix} \tag{2.40}$$

$\mathcal{C}^{14}$ – **Cluster Busy Times** : A cluster busy times constraint limits the number of time groups in which a resource may be busy. The deviation is given by number of cases in which the resource is busy for less than $\underline{B}_c \in \mathbb{N}$ time groups, or for more than $\overline{B}_c \in \mathbb{N}$ time groups. Let $tg \in \mathcal{TG}_c$ denote a time group in which such

a constraint applies:

$$\underline{B}_c - \sum_{tg \in \mathcal{TG}_c} p_{r,tg} \leq s^{14}_{c,r} \quad \begin{matrix} \forall c \in \mathcal{C}^{14} \\ \forall r \in \mathcal{R}_c \end{matrix} \tag{2.41}$$

$$\sum_{tg \in c} p_{r,tg} - \underline{B}_c \leq s^{14}_{c,r} \quad \begin{matrix} \forall c \in \mathcal{C}^{14} \\ \forall r \in \mathcal{R}_c \end{matrix} \tag{2.42}$$

$\mathcal{C}^{15}$ – **Limit Busy Times** : Limit busy times constraint places limits on the number of times a resource may be busy within some time groups. These constraints produce a deviation for each time group. The deviations are given by the number of cases in which the resource is busy for less than $\underline{B}_c \in \mathbb{N}$ time groups, or for more than $\overline{B}_c \in \mathbb{N}$ time groups.

$$\underline{B}_c - |\mathcal{T}_{tg}| \times (1 - p_{r,tg}) - \sum_{t \in \mathcal{T}_{tg}} q_{r,t} \leq s^{15}_{c,r,tg} \quad \begin{matrix} \forall c \in \mathcal{C}^{15} \\ \forall r \in \mathcal{R}_c \\ \forall tg \in \mathcal{TG}_c \end{matrix} \tag{2.43}$$

$$\sum_{t \in \mathcal{T}_{tg}} q_{r,t} - \overline{B}_c - |\mathcal{T}_{tg}| \times (1 - p_{r,tg}) \leq s^{15}_{c,r,tg} \quad \begin{matrix} \forall c \in \mathcal{C}^{15} \\ \forall r \in \mathcal{R}_c \\ \forall tg \in \mathcal{TG}_c \end{matrix} \tag{2.44}$$

$\mathcal{C}^{16}$ – **Limit Workload** : The workload of a resource is given by $W_{e,se,er} = \frac{D_{se} \times L_{er}}{D_e}$, in which $L_{er} \in \mathbb{N}$ is the workload of the event resource er. These values are given as inputs within the information related to events. A limit workload constraint places limits on the total workload that is assigned to resources. The deviation of this constraint, $s^{16}_{c,r}$, is the amount of cases in which the resource workload falls short $\underline{B}_c \in \mathbb{N}$, or exceeds $\overline{B}_c \in \mathbb{N}$, rounded up to the nearest integer.

$$\underline{B}_c - \sum_{e \in \mathcal{E}_c} \sum_{se \in \mathcal{SE}_e} \sum_{t \in \mathcal{T} \setminus \{t_D\}} \sum_{er \in \mathcal{ER}_e} W_{e,se,er} \times x_{se,t,er,r} \leq s^{16}_{c,r} \quad \begin{matrix} \forall c \in \mathcal{C}^{16} \\ \forall r \in \mathcal{R}_c \end{matrix} \tag{2.45}$$

$$\sum_{e \in \mathcal{E}_c} \sum_{se \in \mathcal{SE}_e} \sum_{t \in \mathcal{T} \setminus \{t_D\}} \sum_{er \in \mathcal{ER}_e} W_{e,se,er} \times x_{se,t,er,r} - \overline{B}_c \leq s^{16}_{c,r} \quad \begin{matrix} \forall c \in \mathcal{C}^{16} \\ \forall r \in \mathcal{R}_c \end{matrix} \tag{2.46}$$

2.5.3 Objective Function

Given the slack variables for each XHSTT constraint, the cost of a point-of-application of a constraint $c \in \mathcal{C}$ is defined based on three properties: $type_c$ (it can be either *hard* or *soft*), weight ($w_c \in \mathbb{N}$), and the *CostFunction* (CF) to use.

The cost of a constraint $c \in \mathcal{C}$, which contains slack variable $s_{c,p,d}$, is denoted by $f(s_{c,p,d})$, and it is calculated as shown in (2.47).

$$f(s_{c,p,d}) = w_c \times CostFunction(s_{c,p,d}). \tag{2.47}$$

Three different cost function types are allowed: linear, quadratic and step. These functions are evaluated in terms of slack variables $s_{c,p,d}$, such as follows:

Linear : Sum of deviations.

$$CF^{Linear} = \sum_{p \in \mathcal{P}_c} \sum_{d \in \mathcal{D}_p} s_{c,p,d}. \tag{2.48}$$

Quadratic : Sum of deviation squares.

A variable $s_{c,p,d,i} \in \{0, 1\}$ is introduced to handle with this non-linear cost function. It assumes value 1 if the deviation $d \in \mathcal{D}_p$ of the point of application $p \in \mathcal{P}_c$ of constraint c has the value $i \in \mathcal{I}$, or 0 otherwise. Let $\mathcal{I} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$:

$$CF^{Quadratic} = \sum_{p \in \mathcal{P}_c} \sum_{d \in \mathcal{D}_p} \sum_{i \in \mathcal{I}} i^2 \times s_{c,d,p,i} \tag{2.49}$$

The link from $s_{c,p,d}$ to $s_{c,p,d,i}$ is given by:

$$\sum_{p\in\mathcal{P}_c}\sum_{d\in\mathcal{D}_p}\sum_{i\in\mathcal{I}} i \times s_{c,d,p,i} = s_{c,p,d} \quad \begin{matrix}\forall p\in\mathcal{P}_c \\ \forall d\in\mathcal{D}_p\end{matrix} \tag{2.50}$$

It is also necessary to ensure that only a single integer value is selected by the binary indicator. Therefore the following constraint set must be considered:

$$\sum_{i\in\mathcal{I}} i \times s_{c,d,p,i} = 1 \quad \begin{matrix}\forall p\in\mathcal{P}_c \\ \forall d\in\mathcal{D}_p\end{matrix} \tag{2.51}$$

Step : Penalizes the number of deviations, regardless their magnitudes.

The binary variable $u^{Step}_{c,p,d}$ is introduced. It assumes value 1 iff $s_{c,p,d} > 0$ for constraint c, point-of-application $p \in \mathcal{P}_c$, and deviation $d \in \mathcal{D}_p$, or 0 otherwise. The cost function and constraint set are formulated as shown in (2.52) and (2.53), respectively.

$$CF^{Step} = \sum_{p\in\mathcal{P}_c}\sum_{d\in\mathcal{D}_p} u^{Step}_{c,p,d} \tag{2.52}$$

$$M \times u^{Step}_{cp,d} \geq s_{c,p,d} \quad \begin{matrix}\forall p\in\mathcal{P}_c \\ \forall d\in\mathcal{D}_p\end{matrix} \tag{2.53}$$

in which $M \in \mathbb{N}$ is a sufficiently large number.

Let Ψ_c be the set of tuples that a constraint $c \in C$ applies to. For example, suppose $c \in C^4$, then for a $\tau \in \Psi_c$, $\tau \equiv (e, er)$ and $s^4_{c,\tau} \equiv s^4_{c,e,er}$. Thus, the objective function can be stated as shown in Equation (2.54).

$$\min z = \sum_{j=1}^{16}\sum_{c\in C^j}\sum_{\tau\in\Psi_c} f(s^j_{c,\tau}) \tag{2.54}$$

The full IP model would consist of minimizing z, subject to constraints (2.2) to (2.54). However a different approach is taken. This model is solved in two steps as follows.

By the definition of XHSTT, hard constraints always take priority over soft constraints. However, hard constraints still can be violated, which implies in a hard cost of a solution. Considering this feature, the following approach is taken to solve the model: in Step 1 an Integer Programming model containing only the hard constraints is built. This model is given to the solver, which runs until the time limit is reached or until the model is solved to optimality. The objective value found is the *hard cost* of the solution (hopefully 0). If the IP has been solved to optimality, Step 2 is performed, which consists of adding all the soft constraints to the model and warm-starting the solver from its previous state. Additionally, a constraint is added to ensure that the optimal value of the hard cost is kept. Let z^{hard} denote the sum of the cost of all slack variables of hard constraints, the following constraint is added:

$$z^{hard} = hard\ cost \tag{2.55}$$

Once such an IP model is solved, the cost of the obtained solution, minus the hard cost found in Step 1, is the *soft cost* of the solution. The nature of this solution approach resembles lexicographic multi-objective optimization.

Chapter 3

Formulations

This chapter presents the proposed mathematical programming formulations for XHSTT. Section 3.1 presents several valid inequalities and pre-processing techniques leading to an alternative formulation for XHSTT. Section 3.2 presents a Dantzig-Wolfe column generation approach for educational timetabling. Finally, Section 3.3 presents a cut-and-solve approach applied to XHSTT.

3.1 Alternative Formulation

In this section, starting from the original formulation $\mathscr{F}_1$ presented in the previous section, some valid inequalities and an extended flow based formulation for XHSTT are proposed, yielding an improved formulation denoted throughout the paper as $\mathscr{F}_2$. As can be seen later in Section 5.3, $\mathscr{F}_2$ is stronger than $\mathscr{F}_1$, since it is strictly contained in $\mathscr{F}_1$, i.e. there are valid fractional points for $\mathscr{F}_1$ that are not valid for $\mathscr{F}_2$. These cuts were found either by analysing fractional solutions of $\mathscr{F}_1$ (3.1.5 and 3.1.7) or with the aid of an automatic integer programming reformulation tool [11] (3.1.4 and 3.1.6), which searches for fractional multipliers to generate Chvàtal-Gomory cuts [31]. This is a computationally expensive offline tool. Once these multipliers were discovered, it was relatively easy to generalize these cuts to the timetabling context, such that now their separation is trivial. Pre-processing routines are also proposed to remove unnecessary constraints and variables. All those changes are described in the following subsections.

3.1.1 Generation of Sub-events

In $\mathscr{F}_2$, only feasible sub-events are generated. A sub-event is called feasible when it does not violate any hard split events constraint. This reduction aims to make the resulting IP model smaller, faster to build, and less dependant on the duration of the events. For example, suppose an event e of duration $D_e = 4$ and a hard split events constraints c stating that e should be split into sub-events of duration 2 ($\overline{B}_c^{dur} = \underline{B}_c^{dur} = 2$). In the complete formulation, a set of sub-events with the following lengths 1, 1, 1, 1, 2, 2, 3, and 4 would be generated. In this reduced reformulation, the resulting set of sub-events would be considerably smaller, creating only two sub-events of lengths 2 and 2. However, this benefit is achieved at expense of not ensuring optimality in a special case in which the optimal solution has hard cost different from zero. In practice, this is not a big problem for two reasons: (i) most of XHSTT problems are feasible (i.e. they have at least one possible solution with hard cost equals to zero), and; (ii) in real world problems, it is expected to exist a solution having zero cost of hard constraints violation. Algorithm 1 presents the enhanced procedure to generate sub-events.

The algorithm takes as input the set of events $\mathcal{E}$ and a set of split events constraints $\mathcal{C}^3$. Each event has, initially, an empty set of sub-events $\mathcal{SE}_e$ (line 3). For each possible duration k of sub-events, if the event is not constrained by any hard split events constraint (line 5), the all possible sub-events of duration k are generated normally (lines 6 to 8). Otherwise, sub-events of duration k will be generated only if $k \geq \underline{B}_c^{dur}$ and $k \leq \overline{B}_c^{dur}$ (line 10). In line 14, the sub-events of an event e, $\mathcal{SE}_e$ are added to the full set of sub-events $\mathcal{SE}$.

3.1.2 Alternative Formulation for Link Events

One of the constraints that makes this problem difficult is the requirement of linked events. In this alternative formulation, the structure of link events constraints is explored. When events are connected by hard link events constraints, they must occur at

Algorithm 1: Alternative procedure to generate sub-events in $\mathscr{F}_2$.

Input: A set of events $\mathcal{E}$ and a set of split events constraints $\mathcal{C}^3$.
Output: A set of sub-events $\mathcal{SE}$ per event $e \in \mathcal{E}$.

$\mathcal{SE} = \emptyset$;
foreach $e \in \mathcal{E}$ **do**
  $\mathcal{SE}_e = \emptyset$;
  foreach $k = \{1, \ldots, D_e\}$ **do**
    if $\nexists c \in \mathcal{C}^3 \mid e \in \mathcal{E}_c \ \wedge \ type_c = hard$ **then**
      foreach $j = \{k, \ldots, D_e\}$ **do**
        $D_{se} = k$;
        $\mathcal{SE}_e = \mathcal{SE}_e \cup \{D_{se}\}$;
    else
      if $k \geq \underline{B}_c^{dur}$ **and** $k \leq \overline{B}_c^{dur}$ **then**
        foreach $j = \{k, \ldots, D_e\}$ **do**
          $D_{se} = k$;
          $\mathcal{SE}_e = \mathcal{SE}_e \cup \{D_{se}\}$;
  $\mathcal{SE} = \mathcal{SE} \cup \mathcal{SE}_e$;
return $\mathcal{SE}$;

the same time and they must be split in the same way. Considering this fact, constraint sets (2.28), (2.29), (2.30), and (2.31) can be replaced by:

$$y_{se,t} = y_{\hat{se},t} \qquad \begin{array}{c} \forall\ c \in \mathcal{C}^{10},\ eg \in \mathcal{EG}_c, \\ e \in eg \mid first(eg) = e, \\ \hat{e} \in eg \mid first(eg) \neq \hat{e}, \\ (se,\ \hat{se}) \in (\mathcal{SE}_e,\ \mathcal{SE}_{\hat{e}}),\ t \in \mathcal{T} \setminus \{t_D\} \end{array} \tag{3.1}$$

in which $first(.)$ returns the first element of the set of events in event group eg. When the events that belong to the same event group constrained by link events have different durations or are constrained by different split events constraints, this reformulation is not applied. Once again, such a reformulation does not ensure optimality if the optimal solution has hard cost different from zero.

3.1.3 Alternative Formulation for Avoid Clashes

In real world problems, as well as for all the existing XHSTT instances, the avoid clashes constraints are always hard ones. Therefore, constraint sets (2.4), (2.11), (2.12), and (2.36) can be replaced by:

$$\sum_{se \in \mathcal{SE}} \sum_{er \in \mathcal{ER}_{se}} \sum_{\hat{t} \in \mathcal{T}^{start}_{se,t}} x_{se,\hat{t},er,r} = q_{r,t} \quad \begin{matrix} \forall t \in \mathcal{T} \setminus \{t_D\} \\ \forall r \in \mathcal{R} \end{matrix} \tag{3.2}$$

Note that the auxiliary variable type $v_{t,r} \in \mathbb{N}$ is no longer used in $\mathscr{F}_2$. Consequently, it was removed from the model in the alternative formulation.

3.1.4 Alternative Formulation to Link X and Y (LXY)

The link between variables $x_{se,t,er,r}$ and $y_{se,t}$ in $\mathscr{F}_1$ might be strengthened if one considers the link of each single variable $x_{se,t,er,r}$ to each single variable $y_{se,t}$, instead of linking a set of variables $x_{se,t,er,r}$ to $|er_{se}| \times y_{se,t}$. Furthermore, when a hard prefer resources constraint applies to event e, only the preferred resources should be eligible for the assignment (assuming the existence of a feasible solution for the problem). Taking into account such points, constraint set (2.3) was replaced by:

$$x_{se,t,er,r} = y_{se,t} \quad \begin{matrix} \forall se \in \mathcal{SE} \\ \forall t \in \mathcal{T}, \\ \forall er \in \mathcal{ER}_{se}, \\ \forall r \in \mathcal{R}_{er} \,:\, r \in c \in \mathcal{C}^5 \,\vee\, C^5 = \emptyset \,\vee\, type_c \neq hard \end{matrix} \tag{3.3}$$

3.1.5 Cluster Busy Times Cut (CBT)

A lower bound on the number of days a resource r is busy can be deduced by taking the sum of the durations of the events that are preassigned to r divided by the number of times per day ($timesDay$). This bound is rounded up to the nearest integer. This cut was first proposed by Santos *et al.* [67] and adapted to XHSTT in this work:

$$\sum_{tg \in \mathcal{TG}_c} p_{r,tg} \geq \left\lceil \frac{\sum_{\substack{e \in \mathcal{E}: \\ er \in \mathcal{ER}_e}} D_e}{\frac{\hat{\rho}_{er,r}=1}{timesDay}} \right\rceil \quad \begin{matrix} \forall c \in \mathcal{C}^{14} \\ \forall r \in \mathcal{R}_c \end{matrix} \tag{3.4}$$

3.1.6 Link Y and Q Cut (LYQ)

For any resource r and time t, if r is busy at t, at least one of the sub-events that can be assigned to r will be occurring at t. If a hard prefer resources constraint applies to the sub-event, the sub-event will be considered only when the resource is a preferred resource.

$$q_{r,t} \leq \sum_{\substack{se \in \mathcal{SE}: \\ r \in er \in \mathcal{ER}_{se} \wedge \\ r \in c \in \mathcal{C}^5 \vee \mathcal{C}^5 = \emptyset \vee type_c \neq hard}} y_{se,t} \quad \begin{matrix} \forall r \in \mathcal{R} \\ \forall t \in \mathcal{T} \setminus \{t_D\} \end{matrix} \tag{3.5}$$

3.1.7 Number of Busy Times Cut (NBT)

The number of busy times of a resource r is computed through variables $q_{r,t}$. However, in order to strengthen the formulation, this number could be explicitly given to the IP model when no resource assignment is required:

$$\sum_{t \in \mathcal{T}} q_{r,t} = \sum_{\substack{e \in \mathcal{E}: \\ er \in \mathcal{ER}_e \wedge \hat{\rho}_{er,r}=1}} D_e \quad \forall r \in \mathcal{R} \tag{3.6}$$

If an assign resource constraint is present, for any resource eligible to more assignments than the preassigned ones, the inequality is given by:

$$\sum_{t \in \mathcal{T}} q_{r,t} \geq \sum_{\substack{e \in \mathcal{E}: \\ er \in \mathcal{E}\mathcal{R}_e \wedge \hat{\rho}_{er,r}=1}} D_e \quad {}^{\forall r \in \mathcal{R}:}_{\exists c \in \mathcal{C}^1 \mid type_r = roleType(role_c)} \tag{3.7}$$

3.1.8 Multicomodity Flow Reformulation (MCF)

Dorneles *et al.* [27] modelled the High School Timetabling problem as a Multicomodity Flow problem. The model tackled by Dorneles *et al.* covers only a subset of the features and constraints present in XHSTT. Therefore, an adaptation of Dorneles' model that address the XHSTT features is presented.

Each resource is represented by a commodity. For each resource $r \in \mathcal{R}$ a graph whose flow represents the resource's schedule is created. Let $\mathcal{G}_r = (\mathcal{V}_r, \mathcal{A}_r)$ denote such a graph, in which $\mathcal{V}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. Each node has a set of pull out arcs $\mathcal{A}^+_{r,v} \subseteq \mathcal{A}$ and a set of pull in arcs $\mathcal{A}^-_{r,v} \subseteq \mathcal{A}$. Variables

$$\varphi_a = \begin{cases} 1 & \text{if flow goes through arc } a. \\ 0 & \text{otherwise.} \end{cases}$$

are added to the new formulation $\mathscr{F}_2$ and parameter b_v has value 1 when node v is the source, -1 when it is the sink, or 0 otherwise.

Figure 3.1 presents an example of this graph, in which all types of arcs are shown for a given resource r. Each arc has a specific meaning. Whenever the arc has the same meaning of a variable in $\mathscr{F}_1$, new variable φ_a is not created and the respective variable in $\mathscr{F}_1$ is used instead. The meaning of each type of arc $a \in \mathcal{A}$ is given below:

- Assignment arcs are used to denote that a given resource r is attending one event at time t. These arcs are denoted by binary variables $q_{r,t}$, whose meaning is the same described before in the complete formulation.

- Idle time arcs represent that a resource r has an idle time between busy times at time t in time group tg. These arcs correspond to binary variable $h_{r,tg,t}$, which also came from the original formulation.

- Cluster busy time arcs denote that a given resource r is busy for at least one time in time group tg. These arcs are denoted by binary variables $p_{r,tg}$, which are also the same from the complete formulation.

- Arcs $a^{IN}_{r,tg,t}$ and $a^{OUT}_{r,tg,t}$ are given for each resource r, time group tg, and time t. They denote, respectively, that the first assignment in time group tg for resource r is at time t and that the last assignment is at time t.

- Day-off arcs $a^{OFF}_{r,tg}$ represent that a resource r is not busy at any of the times in time group tg. These arcs lead to a node that represents the next time group or to the sink node.

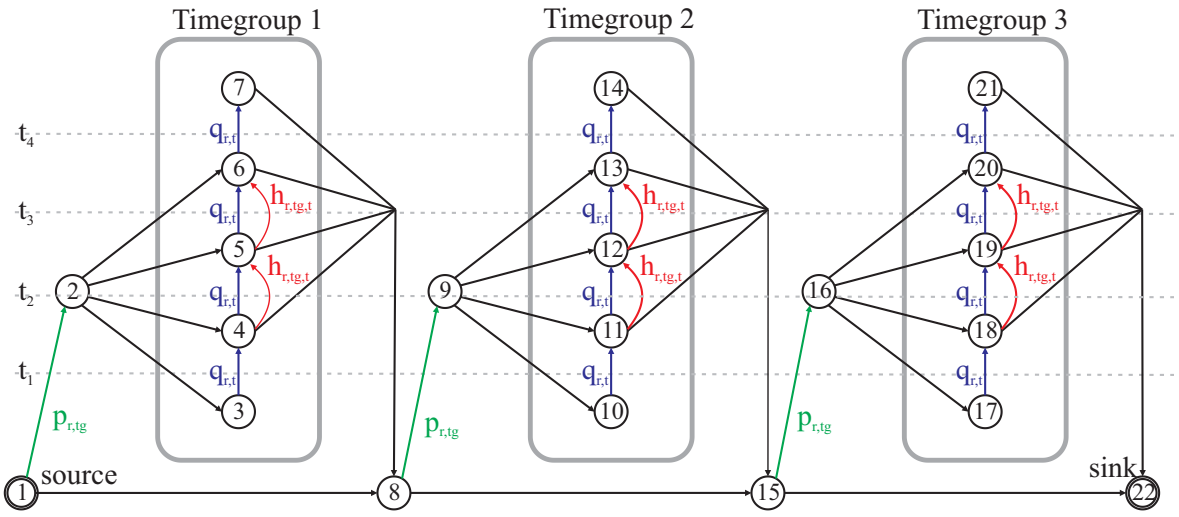


Figure 3.1: Example of network for a resource in a toy instance consisting of three days, having four times each (adapted from [27]).

Every path in such a graph starts from the source node, alternates the arcs providing information about the time assignments, idle times, and busy time groups (usually days)

for the resource and ends at the sink node. Figure 3.2 presents an example of a feasible flow for a given resource. In this example, the resource is busy at the first time (t_1) of Timegroup 1, has a idle time at t_2 and is busy again at times t_3 and t_4. In sequence, the resource has a day off in Timegroup 2 and it is busy again at times t_2 and t_3 in Timegroup 3.

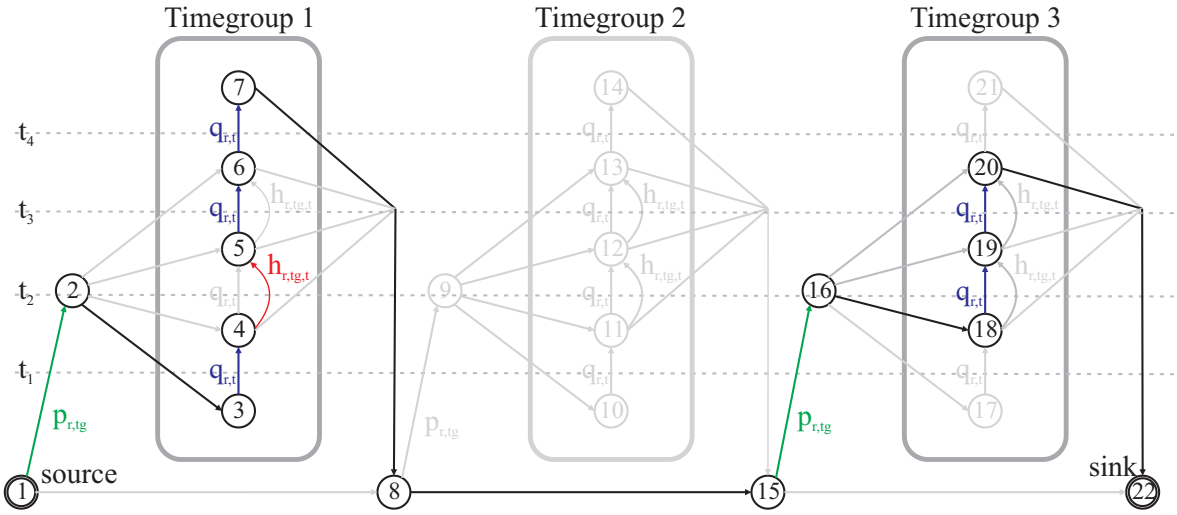


Figure 3.2: Example of schedule for one resource r (adapted from [27]).

The following set of constraints is added to the formulation to ensure the flow conservation in the nodes:

$$\sum_{a \in \mathcal{A}^{+}_{r,v}} \varphi_a - \sum_{a \in \mathcal{A}^{-}_{r,v}} \varphi_a = b_v \quad \begin{matrix} \forall r \in \mathcal{R} \\ \forall v \in V_r \end{matrix} \tag{3.8}$$

Some paths should be explicitly forbidden on the network flow formulation since they lead to miss calculation of penalties. Figure 3.3 illustrates the two cases that shall be avoided. In the first case, an arc goes to node 5 and another leaves right from node 5. This case leads to miss calculation of penalties for Cluster Busy Times constraints because the flow goes into a time group even if it does not have any assignment in any time of that day. Therefore a time group would be counted as busy when it is in fact not busy. Constraint (3.9) is added to forbid such paths:

$$a^{IN}_{r,tg,t} + a^{OUT}_{r,tg,t} \leq 1 \quad \begin{matrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \\ \forall t \in \mathcal{T}_{tg} \end{matrix} \tag{3.9}$$

In the second case, constraint limit idle times is miss calculated because a time should not be counted as idle when there is no busy time before it in a given time group. The time also should not be counted when there is no busy time after itself within the same time group. Constraints (3.10) and (3.11) disable such paths:

$$a^{IN}_{r,tg,t} + h_{r,tg,t} \leq 1 \quad \begin{matrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \\ \forall t \in \mathcal{T}_{tg} \end{matrix} \tag{3.10}$$

$$a^{OUT}_{r,tg,t} + h_{r,tg,t} \leq 1 \quad \begin{matrix} \forall r \in \mathcal{R} \\ \forall tg \in \mathcal{TG} \\ \forall t \in \mathcal{T}_{tg} \end{matrix} \tag{3.11}$$

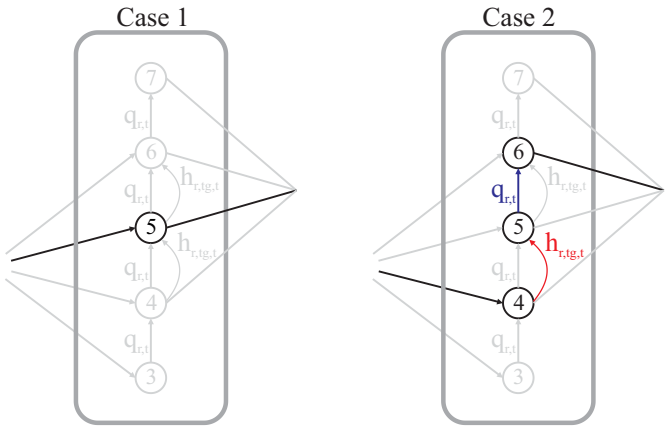


Figure 3.3: Forbidden paths in the network (adapted from [27]).

Constraints cluster busy times and limit idle times are handled by this network flow model. Therefore constraint sets (2.13), (2.14) and (2.38) can be removed from $\mathscr{F}_2$.

3.2 Column Generation

This section presents the application of Dantzig-Wolfe decomposition principles [21] to the alternative formulation. This approach is based on the convexification of extreme points for a subset of constrains in a linear program. The technique works in two steps. In the first step, the original problem is reformulated in such a way that a subset of the constraints is convexified. In the second step, the reformulated problem is solved using column generation. This process is iteratively executed until no column has negative reduced cost. The reformulated version of the original problem, without the refereed subset of constraints, is known as the Master Problem and the problem of calculating the cost of a column through this subset of constraints, as the Pricing Problem. A column can be seen as a possible piece of a solution. For instance, in a vehicle routing problem a column could be formulated as a possible route for a single vehicle and the final solution would be a selection of one column (route) for each vehicle respecting the problem constraints.

In the present application of Dantzig-Wolfe Column Generation, the Pricing Problem consists of finding, for each resource $r \in \mathcal{R}$, the configuration of busy/idle times in its timetable with the smallest reduced cost. In this formulation, each configuration of busy/idle times is a column and will be also refereed to as pattern. Considering this approach, all resource related constraints ($\mathcal{C}^{11}$ - $\mathcal{C}^{16}$) are calculated in the Pricing Problem, leaving only the remaining ones to be handled in the Master Problem. This approach leads to a combinatorial explosion on the number of columns. Therefore, the columns are generated on demand throughout the optimization process. This scheme is known as delayed column generation.

Table 3.1 presents an example of a column for a resource. Suppose there is a penalty of one unity for each idle time between busy times and of five unities for having more than four working days on the week. Hence this column (timetable) have a cost of seven unities.

Table 3.1: Example of pattern representation in the proposed Dantzig-Wolfe decomposition.

Prof. Smith

Mon	Tue	Wed	Thu	Fri
✓	✓	✓	✓	✓
✓	✓		✓	✓
	✓	✓	✓	
✓	✓	✓	✓	
✓	✓	✓	✓	

Binary variables $\lambda_{r,k}$ are introduced to the model for each resource $r \in \mathcal{R}$ and possible pattern $k \in \mathcal{K}$:

$$\lambda_{r,k} = \begin{cases} 1 & \text{if resource } r \text{ follows pattern } k. \\ 0 & \text{otherwise.} \end{cases}$$

Parameter $q_{r,t,k}$ is given as input for each resource $r \in \mathcal{R}$, time $t \in \mathcal{T}$, and pattern $k \in \mathcal{K}$. It has value 1 if resource r is busy at time t in pattern k, and 0 otherwise. Having these additional variables and parameters, the Master Problem $\mathscr{M}$ can be stated as:

$$\min z = \sum_{j=1}^{10} \sum_{c \in C^j} \sum_{\tau \in \Psi_c} f(s^j_{c,\tau}) + \sum_{k \in \mathcal{K}} \sum_{r \in \mathcal{R}} \bar{c}_{r,k} \times \lambda_{r,k} \tag{3.12}$$

Subject to:

$$\sum_{k \in \mathcal{K}} \lambda_{r,k} \times q_{r,t,k} = q_{r,t} \quad \forall t \in \mathcal{T}, r \in \mathcal{R} \tag{3.13}$$

$$\sum_{k \in \mathcal{K}} \lambda_{r,k} = 1 \quad \forall r \in \mathcal{R} \tag{3.14}$$

Constraints 2.2-2.27, 2.32-2.35, 3.1, 3.3 and 3.5 are also included in the Master Problem. Note that the integrality constraint is relaxed for all variables in the Master Problem. Constraints 3.13 link variables $\lambda_{r,k}$ and variables $q_{r,t}$. Constraint set 3.14 ensures convexity for $\lambda_{r,k}$ variables. In Equation 3.12 the cost for penalties of event and event group related constraints is minimized. The last term of the objective function represents the cost of the selected patterns for each resource.

The cost of a pattern is calculated in the Pricing Problem. The Pricing Problem $\mathscr{P}_{r,k}$ of a pattern k for a resource $r \in \mathcal{R}$ is an Integer Program as stated below:

$$\min \sum_{j=11}^{16} \sum_{c \in C^j} \sum_{\tau \in \Psi_c} f(s^j_{c,\tau}) - \pi^{\alpha}_r - \sum_{t \in \mathcal{T}} \pi^{\beta}_{t,r} \times q_{r,t} \tag{3.15}$$

Where π^{α} represents the dual value of convexity constraints 3.14 and π^{β} represents the dual value of constraints 3.13.

The Pricing Problem is subject to constraints 2.37, 2.39-2.46, 3.4 and 3.6-3.11. Integrality constraints apply for all variables in the Pricing Problem.

Given the definitions of the Master Problem, of the Pricing Problem, and of a pattern, Algorithm 2 summarizes the developed implementation of Dantzig-Wolfe Column Generation approach for educational timetabling.

Line 1 loads the Master Problem for the input instance P. In line 2, k is set to 0. k is simply an integer index to differentiate the patterns for each resource $r \in \mathcal{R}$. k is updated at each iteration (line 4). *newColumns* is a binary flag that indicates whether there is at least one new column with reduced cost smaller than 0 or not. Initially *newColumns* is set as false ($\perp$) (line 5) and, if a new column having $\bar{c}_{r,k} < 0$ is found,

Algorithm 2: Implementation of Dantzig-Wolfe Column Generation.

Input: XHSTT instance P and a set of resources $\mathcal{R}$.
Output: Best linear solution s found.

```
1  M ← Load master problem for instance P;
2  k ← 0;
3  repeat
4  |  k ← k + 1;
5  |  newColumns ← ⊥;
6  |  (s, π^α, π^β) ← Solve master problem M;
7  |  foreach r ∈ R do
8  |  |  P_{r,k} ← Load pricing problem for r with (π^α, π^β);
9  |  |  (λ_{r,k}, c̄_{r,k}) ← Solve pricing problem P_{r,k};
10 |  |  if c̄_{r,k} < 0 then
11 |  |  |  M ← M ∪ λ_{r,k};
12 |  |  |  newColumns ← ⊤;
13 until newColumns;
14 return s;
```

newColumns is set as true ($\top$) (line 12) indicating that one more iteration is needed (line 13). Each iteration is composed by lines 4 to 12. At each iteration, the Master Problem $\mathscr{M}$ is solved (line 6) generating a linear primal solution s and dual solutions π^α and π^β. For each resource $r \in \mathcal{R}$, the Pricing Problem $\mathscr{P}_{r,k}$ is loaded solved considering the dual vectors previously found (lines 8 and 9). The output of each pricing problem is a new column $\lambda_{r,k}$ and its reduced cost $\bar{c}_{r,k}$. If this new column has a reduced cost smaller than 0 it is added to the master problem for the next iteration and the flag *newColumns* is set to $\top$ indicating the need of a new iteration. The process stops when, for all resources, no column with reduced cost smaller than 0 could be found in $\mathscr{P}_{r,k}$. The optimal linear solution s is given as output in such a case.

3.3 Cut-and-Solve

The cut-and-solve algorithm was first proposed by Climer and Zhang [18] and applied to the travelling salesman problem. Climer and Zhang also prove optimality and termination for this method. Cut-and-solve search is different from traditional tree search as there is no branching. At each node in the search path, a relaxed problem $\mathscr{P}_R$ and a sparse problem $\mathscr{P}_S$ are solved and a constraint is added to the relaxed problem. The sparse problems provide incumbent solutions. When the constraining of the relaxed problem becomes tight enough, its solution value becomes no better than the incumbent solution value. At this point, the incumbent solution is declared to be optimal. The algorithm can be stopped at any-time as an incumbent solution is found at the root node and continuously updated during the search.

Cut-and-solve enjoys two favourable properties. Since there is no branching, there are no "wrong" sub-trees in which the search may get lost. Furthermore, its memory requirement is negligible. For these reasons, it has potential for problems that are difficult to solve using depth-first or best-first search tree methods [18]. Algorithm 3 presents the developed implementation of the cut-and-solve algorithm for educational timetabling.

In line 1, the sparse problem is defined as a copy of the original integer programming problem. The IP problem in built considering the alternative formulation $\mathscr{F}_2$. In line 2, the relaxed problem $\mathscr{P}_R$ is generated by relaxing the integrality constraint for all variables in the original problem $\mathcal{P}$. In sequence the resulting relaxed problem is solved. In line 5, the variables having reduced cost smaller than α is selected to set $\mathcal{L}$.

In line 6, a constraint stating that the search space is restricted to the variables not belonging to $\mathcal{L}$ (i.e. $\sum_{x \in \mathcal{L}} x = 0$) is added to $\mathscr{P}_S$. The resulting integer program is called sparse problem. After that, the sparse problem is solved. If the best lower bound for the relaxed problem is greater or equal to the optimal integer solution generated at line 8, the solution s is returned and claimed to be optimal (lines 9 and 10). In line

Algorithm 3: Implementation of cut-and-solve algorithm.

Input: Integer programming problem $\mathscr{P}$, obj. function $f(.)$, and minimum reduced cost α.
Output: Best solution s found.

```
1  P_S ← P;
2  P_R ← Relax integrality in P_S;
3  Solve P_R;
4  repeat
5  |  L ← Variables in P_R with reduced cost > α;
6  |  Add constraint Σ_{x∈L} x = 0 to P_S;
7  |  s_R ← Optimal linear solution in P_R;
8  |  s ← Optimal integer solution in P_S;
9  |  if f(s_R) ≥ f(s) then
10 |  |  return s;
11 |  Remove constraint Σ_{x∈L} x = 0 from P_S;
12 |  Add constraint Σ_{x∈L} x ≥ 1 to P_S;
13 |  Add constraint Σ_{x∈L} x ≥ 1 to P_R;
14 until |L| = 0;
15 return s;
```

11, the cut applied at line 6 is removed and the sparse problem is not strict to only variables not in $\mathcal{L}$ any more.

Lines 12 and 13 add the piercing cut to both $\mathscr{P}_R$ and $\mathscr{P}_S$. A piercing cut is a cut that removes at least one feasible integer solution from the solution space of $\mathscr{P}$. Thus, the piercing cut tightens the problem and reduces the size of its solution space. This is ensured by adding a constraint setting the sum of the values of all variables that have reduced cost smaller or equal than α to be equal or greater than one. This procedure is repeated until there is no variable to add to the next piercing cut (i.e. $|\mathcal{L}| = 0$).

Chapter 4

Algorithms

This chapter presents the algorithms implemented in this thesis. Section 4.1 presents the constructive algorithm for generating initial solutions. It is based on several graph algorithms and heuristics. Section 4.2 presents the metaheuristic algorithms developed. First, the neighbourhood structure is presented. Afterwards, Variable Neighbourhood Search (VNS) and Late Acceptance Hill-Climbing (LAHC) based algorithms are proposed. Section 4.3 presents a problem-specific Fix-and-Optimize matheuristic to handle this problem, an enhanced version of such a matheuristic, and a Local Branching algorithm. The Fix-and-Optimize algorithm was also combined with a metaheuristic to compose a hybrid solver.

4.1 Constructive Algorithm

The Kingston High School Timetabling Engine (KHE14) was used as a constructive algorithm. KHE is a platform for handling instances of XHSTT and KHE14 is a version of KHE incorporating several performance enhancements made by its author [43]. It also provides a solver, that was used in this work to generate acceptable initial solutions in short time [44]. KHE generates a solution through three steps: *structural phase*, *time assignment phase*, and *resource assignment phase*. For the sake of brevity, only a brief description of the method is provided in sequence. For additional information, please refer to [43, 44].

At the beginning of the *structural phase*, an initial solution is built with no times or resources assigned. In this step, events are split into sub-events, whose durations depend on split event constraints, and the sub-events (or *meets*) are grouped into sets called *nodes*. Sub-events derived from the same event are nested in the same node. Sub-events whose original events are connected by spread events or avoid split assignment constraints also lie in the same node. Events connected by link event constraints have their meets connected in such a way that the time assigned to one of these meets is also extended to the other connected meets. Each meet also contains a set of times called *time domain*, which defines the times that could be assigned to the meet. *Time domains* are chosen based on preferred time constraints. A meet contains one *task* for each demanded resource in the event that it was derived from. Each task also contains a set of resources of the proper type called *resource domain*. The resources available on *resource domain* are based on preferred resource constraints. Pre-assigned times and resources are also assigned in this step [43].

In *time assignment phase*, a time is assigned for each meet. First, a layer, which is a set of nodes containing meets preassigned to a given resource, is built for each resource in which a hard avoid clashes constraint applies. Then, the layers are sorted in such a way that the hardest layers (layers with less available choices for assignment) come first, and the times are assigned to the meets of each layer, one by one. This assignment is done through a minimum-cost matching between meets of a given layer and times. Each edge of the graph has the cost equals to the objective function impact of this assignment.

Finally, the resources are assigned in the *resource assignment phase*. For each resource type, one of two procedures is executed: (*i*) if the resource assignment for this resource type is constrained by avoid split assignment constraints, a resource packing algorithm is invoked; (*ii*) otherwise, a simple heuristic is used. These two procedures can be described as follows:

- The packing of a resource consists on assigning tasks to the resource in such a way that the solution cost is kept as low as possible. It is accomplished through maximum use of the resource, under its workload limits. The resources are placed in a priority queue, in which more demanded resources are prioritized. At each iteration, a resource is dequeued and processed.

- The simple heuristic consists on assigning the resource that minimizes the objective function for each task, from the most constrained to the least one.

It is possible to estimate the amount of tasks whose resource assignment is impossible through a maximum matching in an unweighed bipartite graph (tasks are demand nodes and resources are supply nodes).

4.2 Metaheuristics

Descriptions of the developed metaheuristics, VNS, LAHC, and proposed variants are presented in this section, along with the six-move neighbourhood structure employed for the metaheuristics.

4.2.1 Neighbourhood Structure

The neighbourhood structure $\mathcal{N}(s)$ is composed of six moves (or neighbourhood functions)[1]. This neighbourhood structure is very similar to the one proposed by the winner of ITC2011 [34, 35], except that the move Permute Resources was removed. This move is computationally expensive and it does not provide significant improvement on the candidate solutions. The functions considered are presented in sequence.

[1] In terms of notation, a neighbourhood function k is denoted by $\mathcal{N}_k(s)$.

1. **Event Swap (ES):** Two events e_1 and e_2 are selected and their timeslots t_1 and t_2 are swapped. Table 4.1 presents an example of this move.

Table 4.1: Example of Event Swap neighbourhood.

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

ES $(Eng_1, Span_1)$ $\Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Span_1 *Gary*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Span_1 *Gary*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Eng_1 *John*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Eng_1 *John*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

2. **Event Move (EM):** An event e_1 is moved from its original timeslot t_1 to a new empty timeslot t_2. Table 4.2 presents an example of this move.

Table 4.2: Example of Event Move neighbourhood.

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

EM $(Geog_2, Fri_5)$ $\Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_2 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*		Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	Geog_2 *Mark*

3. **Event Block Swap (EBS):** Similarly to ES move, the Event Block Swap swaps the timeslots of two events e_1 and e_2. However, if the events have different durations, e_1 is moved to the last timeslot occupied by e_2. This move allows timeslot swaps without losing the allocation contiguity. Table 4.3 presents an example of this move.

Table 4.3: Example of Event Block Swap neighbourhood.

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

EBS $(Math_2, Geog_2)$ $\Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_2 *John*	Geog_2 *Mark*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Math_2 *Anna*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

4. **Resource Swap (RS):** Two resources r_1 and r_2, assigned to events e_1 and e_2, are swapped. Such an operation is only allowed if the resources r_1 and r_2 are of the same type (i.e. both have to be teachers). Table 4.4 presents an example of this move.

Table 4.4: Example of Resource Swap neighbourhood.

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	

RS $(Chem_1, Phis_1)$ $\Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Sara*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Sara*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Paul*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Paul*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Paul*	His_1 *Mark*	His_2 *Mark*	

5. **Resource Move (RM):** The resource r_1, assigned to an event e_1, is replaced by a new resource r_2, randomly selected from the available resources that can be used to attend e_1. Table 4.5 presents an example of this move.

Table 4.5: Example of Resource Move neighbourhood.

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_1 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Gary*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Gary*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	
Compatible:		Tom	Joey	Kate

RM $(Span_1, Kate) \Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Math_1 *Anna*	Eng_2 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Eng_1 *John*	Math_2 *Anna*	Phis_1 *Paul*	Eng_2 *John*
Math_1 *Anna*	Chem_1 *Sara*	Geog_2 *Mark*	Span_1 *Kate*	Eng_2 *John*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	Span_1 *Kate*	Phis_2 *Paul*
Geog_1 *Mark*	Chem_1 *Sara*	His_1 *Mark*	His_2 *Mark*	
Compatible:		Tom	Joey	Gary

6. **Kempe Move (KM):** Two timeslots t_1 and t_2 are selected. The events assigned to t_1 and t_2 are listed and represented as nodes in a graph. If two nodes (events) n_1 and n_2 share resources, they are connected by an edge. Edges are created only between nodes assigned in distinct timeslots. Therefore, the generated graph is bipartite, and it is known as conflict graph. Every edge in the conflict graph has a weight, which is the cost difference in the objective function assuming the exchange of timeslots between the events in the pair (n_1, n_2); and, obviously, subtracting the temporary penalty for the produced clash. Afterwards, the method looks for the lowest cost path in the conflict graph and it makes the exchange of timeslots in the chain. This procedure is similar to the proposed by Tuga *et al.* [74]. Figure 4.1 presents an example of this move.

When generating a random neighbour, the neighbourhood to be considered is chosen according to the following probabilities: if the instance requires the assignment of resources (i.e. there is at least one Assign Resource constraint), the probabilities are: ES = 0.20, EM = 0.38, EBM = 0.10, RS = 0.20, RM = 0.10, and KM = 0.02. Otherwise, the neighbourhoods RS and RM are not used and the probabilities are: ES = 0.40, EM = 0.38, EBS = 0.20, and KM = 0.02. These probabilities were empirically adjusted. Higher probabilities were given to moves that take a short time and have a good potential to improve the solution, while smaller probabilities were set to the ones that take more processing time.

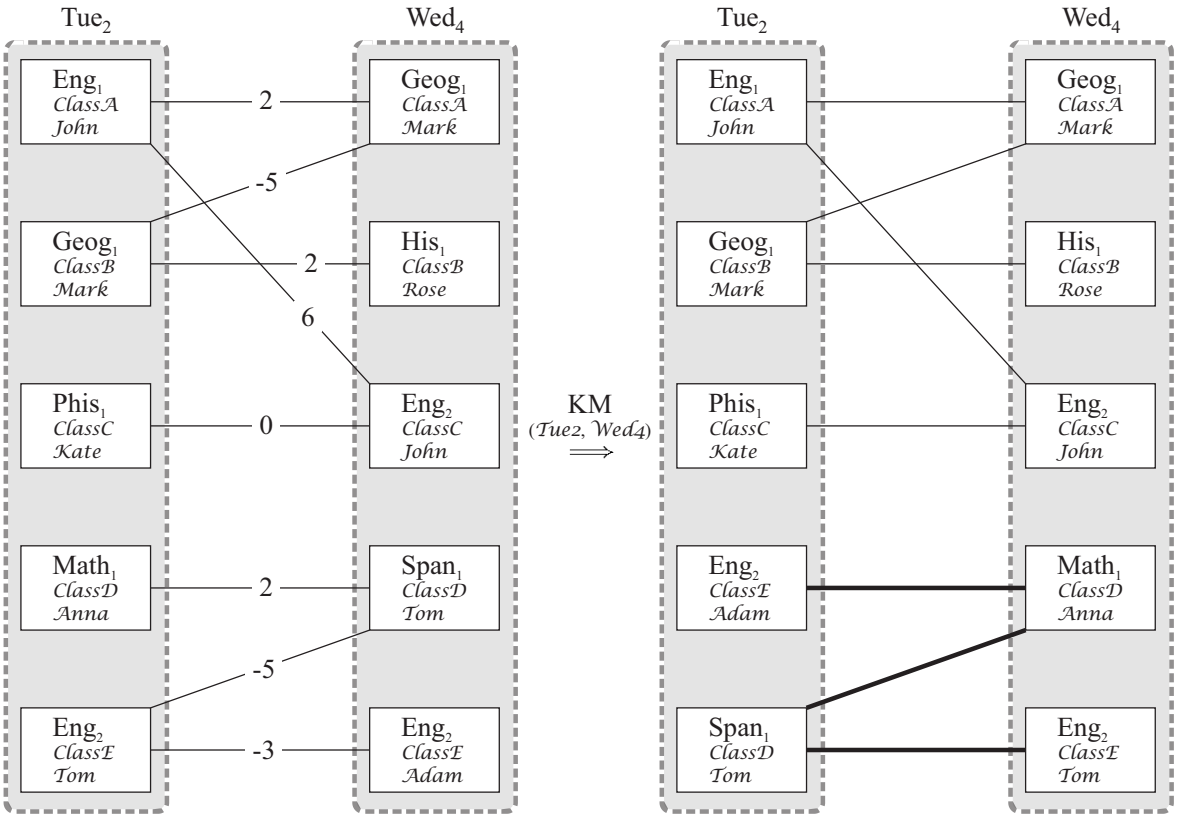


Figure 4.1: Example of Kempe Move.

4.2.2 Variable Neighbourhood Search

The Variable Neighbourhood Search algorithm was proposed by Mladenovic and Hansen [52] and it consists in a local search method that explores the search space by making systematic changes in the neighbourhood structures.

At each iteration, a neighbourhood structure k is selected according to the order presented in Section 4.2.1. A random neighbour $s^{'} \in \mathcal{N}_k(s)$ is generated in this neighbourhood. Afterwards, a descent method is applied to $s^{'}$. If the best solution found by the descent method, $s^{''}$, is better than the best known solution, it is updated and the neighbourhood structure is set to the first one again. Otherwise, the search continues in the next neighbourhood structure. When the last neighbourhood structure ($k_{max} = 6$)

is explored, the search goes back to the first neighbourhood. This process continues until a stop condition is met.

The descent method is a key component of the VNS algorithm. The ability to quickly reach good local optima is critical to the success of the algorithm. Random Non-Ascendent (RNA) movements were employed as the descent method, with the stopping criterion of $Iter_{max} = 1,000,000$ iterations without improvement. This value was adjusted empirically. Algorithm 4 presents the implementation of the RNA method.

Algorithm 4: Developed implementation of RNA.

Input: Initial solution s, neighborhood function $\mathcal{N}(.)$, and obj. function $f(.)$.
Output: Best solution s found.

```
1 Iter ← 0;
2 while Iter < IterMax and elapsedTime < timeLimit do
3     Iter ← Iter + 1;
4     Generate a random neighbour s' ∈ N(s);
5     if f(s') ≤ f(s) then
6         s ← s';
7         if f(s') < f(s) then
8             Iter ← 0;
9 return s;
```

Finally, Algorithm 5 presents the basic implementation of VNS developed. The stop condition adopted is a time limit. Some variants of VNS were implemented and they are presented in the next subsections.

Reduced Variable Neighbourhood Search

A reduction to the original Variable Neighbourhood Search Method was also proposed by Mladenovic and Hansen [52], in which there is no descent phase (Algorithm 5, line 5) to improve the generated solution s' at each iteration. This may improve the VNS performance in cases in which the complete exploration of the defined neighbourhoods is excessively expensive. This reduction is called Reduced Variable Neighbourhood Search (RVNS).

Algorithm 5: Developed implementation of VNS.

Input: Initial solution s, neighborhood function $\mathcal{N}(.)$, and obj. function $f(.)$.
Output: Best solution s found.

```
1  while elapsedTime < timeLimit do
2      k ← 1;
3      while k ≤ k_max do
4          Generate a random neighbour s' ∈ N_k(s);
5          s'' ← descentMethod(s') (Alg. 4);
6          if f(s'') ≤ f(s) then
7              s ← s'';
8              k ← 1;
9          else
10             k ← k + 1;
11 return s;
```

Sequential Variable Neighbourhood Descent

Another variation of the original VNS method is the Sequential Variable Neighbourhood Descent (SVND) [39]. The main difference between SVND and the original VNS is that it explores only a subset of the available neighbourhood structure at each iteration of the descent phase, instead of the whole set. In the present implementation of SVND, the descent method at each iteration is executed considering only one neighbourhood structure k at a time ($s'' \leftarrow descentMethod_k(s')$).

Skewed Variable Neighbourhood Search

If very large neighbourhoods are considered in VNS, the information related to the current best optima dissolves, and the algorithm degenerates into multistart [38]. The Skewed Variable Neighbourhood Search (SVNS) was proposed to handle with such a problem. In this variant, it is adopted a relaxed rule to accept the candidate solution s''. This relaxed rule can accept a worse solution based on its distance to the current incumbent, as follows: $f(s'') - \alpha \times \rho(s, s'')$, in which $\rho(s, s'')$ is the distance between s and s'', and α is a parameter to be set. To compute the distance between two solutions,

the following metric was adopted: for each solution, a string with n positions is built (n is the number of events). In each position i, there is an ordered pair indicating the meeting and tasks which are associated to event i. Then, $\rho(s, s'')$ is the Hamming distance between these two strings. α was empirically set to 1.0.

4.2.3 Late Acceptance Hill-Climbing

The Late Acceptance Hill-Climbing metaheuristic was recently proposed by Burke and Bykov [12, 14]. This algorithm is an adaptation of the classical Hill-Climbing method. It relies on comparing a new candidate solution with the last l-th solution considered in the past, in order to accept or to reject it. Note that the candidate solution may be accepted even if it is worse than the current solution since it is compared to the best solution of l iterations before.

This metaheuristic was created with three goals in mind: (*i*) to be an one-point search procedure that does not employ an artificial cooling schedule, like Simulated Annealing; (*ii*) to effectively use the information collected during previous iterations of the search, and; (*iii*) to employ a simple acceptance mechanism (i.e. almost as simple as Hill-Climbing) [13]. Some successful examples of application of LAHC can be found in [2, 59, 76].

In this method, a vector $\vec{p} = [\vec{p}_0, \ldots, \vec{p}_{l-1}]$ with costs of previous solutions is stored. Initially this list is filled with the cost of the initial solution s: $\vec{p}_k \leftarrow f(s)\ \forall k \in \{0,\ ...,\ l-1\}$. At each iteration i, a candidate solution s' is generated. The candidate solution is accepted if its cost is less or equal to the cost stored on the $i \mod l$ position of $\vec{p}$. Moreover, if this solution is better than the best solution s^* found so far, a new incumbent solution is stored. Afterwards, the position $v = i \mod l$ of $\vec{p}$ is updated: $\vec{p}_v \leftarrow f(s')$. This process repeats until a stopping condition is met. The developed implementation of the LAHC is illustrated in Algorithm 6. Note that time limit was considered as the stopping condition for the algorithm. Parameter l was set to 500 for all experiments using LAHC. This value was empirically adjusted.

Algorithm 6: Developed implementation of LAHC.

Input: Initial solution s, fitness list size l, neighborhood function $\mathcal{N}(.)$, and obj. function $f(.)$.
Output: Best solution s^* found.

```
1  p_k ← f(s) ∀k ∈ {0, ..., l - 1};
2  s* ← s;
3  i ← 0;
4  while elapsedTime < timeLimit do
5      Generate a random neighbour s' ∈ N(s);
6      v ← i mod l;
7      if f(s') ≤ p_v then
8          s ← s';
9          if f(s) < f(s*) then
10             s* ← s;
11     p_v ← f(s);
12     i ← i + 1;
13 return s*;
```

Since it is relatively recent, variants of the LAHC metaheuristic were not extensively explored yet. Therefore, the combination of LAHC with other methods and strategies is an open field for experimentation [13]. Two new variants of LAHC are proposed in this work: Stagnation Free LAHC (SF-LAHC), and Simulated Annealing - Stagnation Free LAHC. These heuristics are described below.

Stagnation Free LAHC

In late stages of the LAHC, it is often hard to improve the current solution since the algorithm usually has a list with all l positions occupied by the same cost value. This behavior can make the LAHC incapable of escaping from local optima, because worse solutions are never accepted. A new variation of the LAHC, so-called Stagnation Free LAHC or simply SF-LAHC, is proposed to handle such situations.

In SF-LAHC method, the algorithm reheats the system when it reaches a stagnation condition. In the proposed implementation, the reheat consists of retrieving the vector of costs from the last time in which one improvement occurred, denoted by $\vec{p}\,'$. It

means that various worsening moves may become acceptable after this list update. The algorithm is considered on stagnation when it performs n iterations without improvement. The author suggests to set n as a function of l, in order to simplify the parameter tuning process. The Stagnation Free variant of LAHC is presented in Algorithm 7. In the experiments, it was set $n = 1,000 \times l$.

Algorithm 7: Proposed Stagnation Free LAHC approach.

Input: Initial solution s, fitness list size l, neighborhood function $\mathcal{N}(.)$, and obj. function $f(.)$.
Output: Best solution s^* found.

```
1  p_k ← f(s) ∀k ∈ {0, ..., l - 1};
2  p_k ← p_k';
3  s* ← s;
4  i ← 0;
5  n ← 1,000 × l;
6  while elapsedTime < timeLimit do
7      Generate a random neighbour s' ∈ N(s);
8      v ← i mod l;
9      if f(s') ≤ p_v then
10         s ← s';
11         if f(s) < f(s*) then
12             s* ← s;
13             p' ← p;
14             i ← 0;
15     p_v ← f(s);
16     i ← i + 1;
17     if i = n then
18         p ← p';
19         i ← 0;
20 return s*;
```

Simulated Annealing - SF-LAHC

Proposed by Kirkpatrick *et al.*[45], the metaheuristic Simulated Annealing (SA) is a probabilistic method based on an analogy to thermodynamics, simulating the cooling of a set of heated atoms. This technique starts its search from an initial solution. The main

procedure consists of a loop that randomly generates, at each iteration, one neighbour s' of the current solution s. Movements are probabilistically selected considering a temperature T and the cost variation obtained with the move, Δ.

This algorithm was part of all ITC winner solvers [34, 46, 54]. Therefore, besides being explored as a standalone solver, it was evaluated in a composed approach with the SF-LAHC algorithm. Since Simulated Annealing performance is not strongly affected by the fitness of the initial solution, it was proposed a mixed algorithm, with the Simulated Annealing algorithm being applied to the initial solution, and the SF-LAHC method being executed further, to refine the solution obtained by SA.

The implementation of Simulated Annealing used in this work is described in Algorithm 8. Parameters were empirically set as $\alpha = 0.97$, $T_0 = 1$ and $SAmax = 10,000$.

Algorithm 8: Developed implementation of SA.

Input: Initial solution s, max. iterations for each temp. $SAmax$, cooling rate α, initial temp. T_0, neighborhood function $\mathcal{N}(.)$, and obj. function $f(.)$.
Output: Best solution s^* found.

```
1  s* ← s;
2  IterT ← 0; T ← T0;
3  while elapsedTime < timeLimit do
4      while IterT < SAmax do
5          IterT ← IterT + 1;
6          Generate a random neighbour s' ∈ N(s);
7          Δ = f(s') − f(s);
8          if Δ < 0 then
9              s ← s';
10             if f(s') < f(s*) then
11                 s* ← s';
12         else
13             Take x ∈ [0, 1];
14             if x < e^(−Δ/T) then
15                 s ← s';
16     T ← α × T;
17     IterT ← 0;
18 return s*;
```

4.3 Matheuristics

Matheuristics are heuristic algorithms made by the cooperation between metaheuristics and mathematical programming methods (MP) [10, 50, 63]. Matheuristic strategies are often classified into hard or soft fixation of variables. In the hard-fixation strategy, a set of variables is fixed to their value in the incumbent solution and another set is optimized per iteration. In the soft-fixation strategy, an additional constraint states that at most a number of variables may change their value per iteration [20]. Both fixation strategies are explored in this thesis. Fix-and-Optimize is considered for hard-fixation of variables and Local Branching for soft-fixation. A variation of the Fix-and-Optimize strategy is also proposed in this thesis.

4.3.1 Fix-and-Optimize

In the Fix-and-Optimize (FixOpt) approach, a metaheuristic works at the master level, controlling low level local search procedures. These local searches are reduced Mixed Integer Programming (MIP) models, in which a subset of variables is fixed to their current values in the incumbent solution, and the remaining variables of the model can be freely modified by the MIP solver.

In the present implementation, the main procedure is similar to the proposed by Sørensen and Stidsen [71]. Considering that $\mathcal{X}$ represents the set of decision variables and s represents the incumbent solution, the proposed Fix-and-Optimize approach is presented in Algorithm 9.

Algorithm 9: Proposed problem-specific Fix-and-Optimize approach.

Input: XHSTT instance P, initial solution s, number of resources n, set of resources $\mathcal{R}$, and maximum optimal iterations in a row $optInRow$.
Output: Best solution s found.

```
1  M ← Load MIP model for instance P and solution s;
2  X ← Load values for variables from s;
3  optIter ← 0;
4  while elapsedTime < timeLimit do
5      V ← ∅;
6      count ← 0;
7      while count < n do
8          Randomly select a resource ρ ∈ R;
9          foreach variable x_{se,t,er,r} ∈ X do
10             if r = ρ then
11                 V ← V ∪ {x_{se,t,er,r}};
12         count ← count + 1;
13     Fix variables X \ V in M to their current value;
14     (s, status) ← Solve M with short time limit;
15     Release fixed variables in M;
16     if status = "Optimal" then
17         optIter ← optIter + 1;
18         if optIter = optInRow then
19             n ← n + 1;
20             optIter ← 0;
21     else if status = "TimeLimit" and n > 1 then
22         n ← n − 1;
23 return s;
```

The algorithm takes as input an instance of XHSTT problem P and an initial solution s (lines 1 and 2). In line 3, $optIter \leftarrow 0$ counts the number of consecutive iterations achieving the optimal solution. In line 5, an empty set ($\mathcal{V}$) of variables is created. While the number n of resources was not fulfilled yet, a resource ρ is randomly selected (line 8). Then, for each variable $x_{se,t,er,r} \in \mathcal{X}$, if the resource r in $x_{se,t,er,r}$ is the same as selected resource ρ, variable $x_{se,t,er,r}$ is added to $\mathcal{V}$ set (lines 10 and 11). In line 13, all variables except the selected ones have their values fixed. In line 14, the MIP solver is invoked with a short time limit. In sequence, all variables are unfixed for the next iteration of the algorithm. Lines 16 to 22 are related to the automatic adjust of n value throughout the optimization process. If $optIter$ reaches the maximum number of consecutive iterations achieving the optimal solution (given by parameter $optInRow$), n is increased by one (line 19). Else if the solver could not finish due to the given time limit (line 21) the number of resources n is reduced by one (line 22).

To perform the computational experiments, the initial number of resources to be selected per iteration was empirically set as $n = 5$. Although an initial value for n has to be provided, n is automatically adjusted throughout the optimization process. The maximum of iterations in a row reaching the optimal solution before increasing n, $optInRow$, was also set as 5. Thus, whenever five consecutive iterations of the Fix-and-Optimize algorithm return the optimal solution, n will be increased by one. For line 13, a short limit of one twentieth of the total available time was considered for each iteration. Usually, the IP solver takes only a few seconds per iteration.

Table 4.6 presents an example of this MIP neighbourhood. Suppose a tiny timetabling problem with only one class (Class A) and three teachers (John, Anna, and Kate). Consider also that it is not desirable that teachers were involved in lectures at more than two days. In this example, resources Anna and Kate are randomly selected to have their schedule optimized in one iteration of Fix-and-Optimize. Coloured events indicate that they are unlocked to be optimized by the IP solver. Anna's slots are blue while Kate's are red. On the left side of the image one can see that both Anna and Kate are involved

in lectures at three days. On the right side it is presented the resulting solution after solving the IP model with all variables related to resources Anna and Kate unfixed. Note that it would be hard to remove this constraint violation through the neighbourhoods considered in this work (Section 4.2.1): it would be necessary a large, and lucky, chain of Event Swap moves.

Table 4.6: Example of one iteration of the proposed Fix-and-Optimize approach.

Class A

Mon	Tue	Wed	Thu	Fri
Eng_1 *John*	Span_2 *John*	Math_1 *Anna*	Chem_1 *Kate*	Math_2 *Anna*
Eng_1 *John*	Span_2 *John*	Math_1 *Anna*	Chem_1 *Kate*	Phi_1 *Kate*
Span_1 *John*	Eng_2 *John*	Bio_1 *Kate*	Prog_1 *Anna*	Math_3 *Anna*
Span_1 *John*	Eng_2 *John*	Bio_1 *Kate*	Prog_1 *Anna*	Phis_2 *Kate*

Fix-Opt $\{Anna, Kate\}$ $\Longrightarrow$

Class A

Mon	Tue	Wed	Thu	Fri
Eng_1 *John*	Span_2 *John*	Math_1 *Anna*	Math_2 *Anna*	Chem_1 *Kate*
Eng_1 *John*	Span_2 *John*	Math_1 *Anna*	Math_3 *Anna*	Chem_2 *Kate*
Span_1 *John*	Eng_2 *John*	Bio_1 *Kate*	Prog_1 *Anna*	Phi_1 *Kate*
Span_1 *John*	Eng_2 *John*	Bio_1 *Kate*	Prog_1 *Anna*	Phis_2 *Kate*

4.3.2 Defect-Oriented Fix-and-Optimize

A variation of the fix-and-optimize algorithm is also proposed in this work. This variation, here named Defect-Oriented Fix-and-Optimize (DO-FixOpt), focuses on smarter ways of selecting the variables to be freed at each iteration of the fix-and-optimize process.

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is created, in which the set of vertex $v \in \mathcal{V}$ represents the resources ($\mathcal{V} \equiv \mathcal{R}$) and the set of edges $e \in \mathcal{E}$ represents the relation between these resources. If two resources r_1 and r_2 share at least one preassigned event in common, there will be an edge $(r_1, r_2) \in \mathcal{E}$ connecting them. These edges also have weights $w_e \in \mathbb{N}$, indicating how strong is the relation between the resources they connect. w_e is equal to the sum of the durations of all preassigned events that r_1 and r_2 have in common. Each vertex is associated to an integer value $b_r \in \mathbb{N}$ that indicates the objective function penalty related to the resource r. This value also takes into account penalties related to the events (and their event groups) that r is preassigned to. To promote some randomness to this process, a parameter $\underline{b} \in \mathbb{N}$ is added to b_r.

Figure 4.2 presents an example of such a graph, in which the size of the vertex is directly proportional to b_r, and w_e is represented on each edge. In this example, it is easy to note that r_2 and r_9 have higher probability of being selected. Given that r_9 is selected, for having shared events whose total duration is 10, r_6 is more likely to be selected together with r_9 than r_7 and r_8.

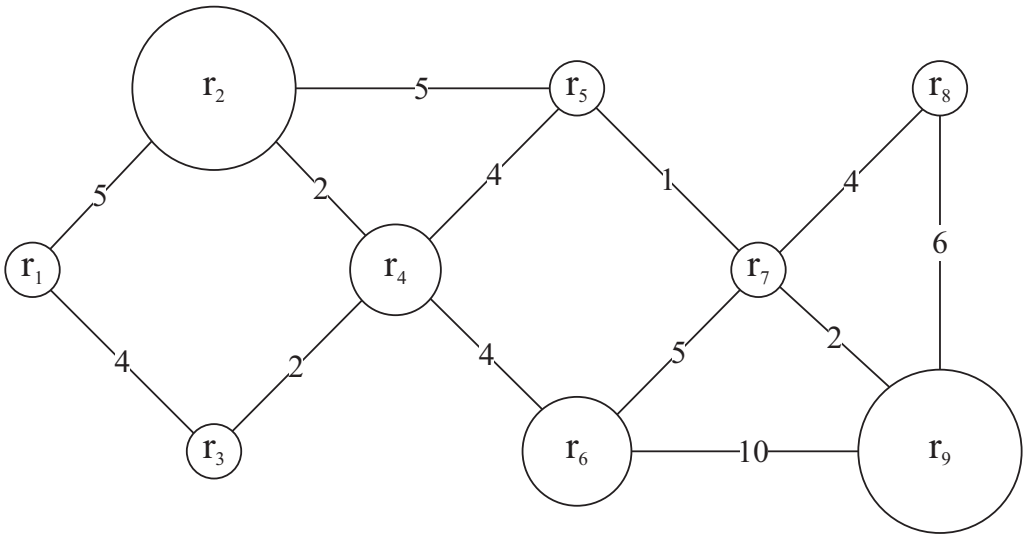


Figure 4.2: Example of graph of resources for the Defect-Oriented Fix-and-Optimize algorithm.

Having such a graph, instead of randomly selecting a resource as it is done in line 8 of Algorithm 9, the resources will be selected based on b_r and w_r. This selection is performed through a roulette wheel method (as higher b_r is, more likely is r to be selected). Whenever a vertex (resource) is selected, the resources that it is connected to are also selected. If the number of adjacent vertices is not enough to fulfill the number of resources to be selected ($|adj_r| \leq n - count$), all adjacent vertex are selected; otherwise, a tournament selection is applied to select the remaining number of resources ($n - count$). In the tournament, the weight w_r of the edges is compared to decide which resource to take.

Parameter $\underline{b}$ plays an important role since it controls the randomness of the resource selection process. On the one hand, if $\underline{b} \to \infty$, the selection is completely randomized (just as the standard Fix-and-Optimize). On the other hand, if $\underline{b} = 0$ the resources having heavier violations will be selected more often than the others. Furthermore, resources with no violations will never be selected. The problem that arises when $\underline{b} = 0$ is that a resource may have a heavy penalty but it might be not possible to fix it. Having this in mind, $\underline{b}$ was set to 1 for all the computational experiments in this thesis.

Algorithm 10 presents the pseudo-code of the proposed Defect-Oriented Fix-and-Optimize approach for timetabling. The main difference from Algorithm 10 to 9 is the addition of lines 10 to 17 to select more efficiently the resources to have their variables released. It is important to highlight that this concept can be easily extended to other combinatorial optimization problems.

4.3.3 Local Branching

The Local Branching (LB) strategy was proposed by Fischetti and Lodi [30]. It is based on the soft fixation of variables in a Mixed Integer Programming model. The soft fixation of variables is achieved by, given an initial solution, adding to the model a constraint stating that at least $k\%$ of the variables must keep its current value. Therefore, a fraction of the model is fixed but there is no constraint regarding which variables shall be kept unchanged. Suppose one is given a 0-1 solution $\bar{s}$ of a MIP problem $\mathscr{P}$ with n variables and at least 90% of the variables must keep their values. The soft fixing constraint is:

$$\sum_{j=1}^{n} \bar{s}_j \times s_j \geq \lceil 0.9 \sum_{j=1}^{n} \bar{s}_j \rceil \tag{4.1}$$

Let $\mathcal{B}$ denote the set of binary variables of a MIP problem. Then $\bar{\mathcal{S}} := \{j \in \mathcal{B} : \bar{s}_j = 1\}$ is the binary support of a solution $\bar{s}$. For a given integer parameter k, the k-OPT

Algorithm 10: Proposed Defect-Oriented Fix-and-Optimize approach.

Input: XHSTT instance P, init. solution s, num. of resources n, set of resources $\mathcal{R}$, max. optimal iterations in a row $optInRow$, and min. artificial cost $\underline{b}$.

Output: Best solution s^* found.

$s^* \leftarrow s$;
$\mathscr{M} \leftarrow$ Load MIP model for instance P and solution s;
$\mathcal{X} \leftarrow$ Load values for variables from s;
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}, b, w\} \leftarrow$ Generate graph of resources considering $\underline{b}$;
$optIter \leftarrow 0$;
while $elapsedTime < timeLimit$ **do**
    $count \leftarrow 0$;
    $\mathcal{R}' \leftarrow \emptyset$;
    while $count < n$ **do**
        Select a resource $\rho \in \mathcal{R}$ in a roulette wheel according to b_ρ ;
        $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{\rho\}$;
        if $|adj_\rho| \leq n - count$ **then**
            $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{adj_\rho\}$;
        else
            $\mathcal{R}^{adj} \leftarrow$ Select $n - count$ resources in $|adj_\rho|$ by tournament;
            $\mathcal{R}' \leftarrow \mathcal{R}' \cup \mathcal{R}^{adj}$;
        $count \leftarrow count + 1$;
    $\mathcal{V} \leftarrow \emptyset$;
    foreach *variable* $x_{se,t,er,r} \in \mathcal{X}$ **do**
        foreach $r' \in \mathcal{R}'$ **do**
            if $r = r'$ **then**
                $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{se,t,er,r}\}$;
    Fix variables $\mathcal{X} \setminus \mathcal{V}$ in $\mathscr{M}$ to their current value;
    $(s, status) \leftarrow$ Solve $\mathscr{M}$ with short time limit;
    Release fixed variables in $\mathscr{M}$;
    if $f(s) < f(s^*)$ **then**
        $s \leftarrow s^*$;
        Update $\mathcal{G}$ according to the violations in s^*;
    if $status =$ "*Optimal*" **then**
        $optIter \leftarrow optIter + 1$;
        if $optIter = optInRow$ **then**
            $n \leftarrow n + 1$;
            $optIter \leftarrow 0$;
    else if $status =$ "*TimeLimit*" **and** $n > 1$ **then**
        $n \leftarrow n - 1$;
return s^*;

neighbourhood $\mathcal{N}(\bar{s}, k)$ is defined as the set of feasible solutions of $\mathscr{P}$ satisfying the additional local branching constraint:

$$\Delta(s, \bar{s}) := \sum_{j \in \bar{\mathcal{S}}} (1 - \bar{s}_j) + \sum_{j \in \mathcal{B} \setminus \bar{\mathcal{S}}} s_j \leq k \tag{4.2}$$

where the two terms in left-hand side count the number of binary variables flipping their value (with respect to $\bar{s}$) either from 1 to 0 or from 0 to 1, respectively.

Since, in the considered IP model for XHSTT, the cardinality of the binary support of any feasible solution of $\mathscr{P}$ is a constant, this constraint was more conveniently written in its equivalent "asymmetric" form

$$\Delta(s, \bar{s}) := \sum_{j \in \bar{\mathcal{S}}} (1 - \bar{s}) \leq k \tag{4.3}$$

The idea is that the neighbourhood $\mathcal{N}(\bar{s}, k)$ corresponding to the left branch must be "sufficiently small" to be optimized within short time, but still "large enough" to likely contain better solutions than s.

Fischetti and Lodi [30] also proposed a heuristic scheme based on local branching. Algorithm 11 presents the developed implementation of a local branching based heuristic for educational timetabling.

Line 1 adds a branching constraint to the problem $\mathscr{P}$. In line 3, the time limit for the solver is set according to the given parameter. In line 4, the problem $\mathscr{P}$ is solved. Since a feasible solution is taken as input, only two status can arrive from the solver:

OPTIMAL : The model has been solved to optimality within the given $timeLimitIt$. In such a case, if the objective function of the solution found is better than the overall best (line 6), the best solution is updated (lines 7 and 8); otherwise, parameter k is increased (line 10) and, if diversification is active, the solution

Algorithm 11: Implementation of local branching based search.

Input: Integer programming problem $\mathscr{P}$, initial solution s, neighborhood size k, and $timeLimitIt$ per iteration.
Output: Best solution s^* found.

```
1  Add branching constraint Δ(s, s̄) ≤ k to 𝒫;
2  while elapsedTime < timeLimit do
3      Set the time limit of the solver to timeLimitIt;
4      (s, status) ← Solve 𝒫;
5      if status = "Optimal" then
6          if f(s) < f(s*) then
7              best ← f(s);
8              s* ← s;
9          else
10             k ← k + ⌈k/2⌉;
11             if diversify then
12                 Set solution limit to one to the next iteration;
13                 diversify ← false;
14             else
15                 diversify ← true;
16         Reverse last local branching const. into Δ(s, s̄) ≥ k + 1;
17         s̄ ← s;
18         Add branching constraint Δ(s, s̄) ≤ k to 𝒫;
19     else if status = "TimeLimit" then
20         if f(s) < f(s*) then
21             s* ← f(s);
22             Remove last local branching constraint from 𝒫;
23         else
24             k ← k − ⌈k/2⌉;
25         s̄ ← s;
26         Add branching constraint Δ(s, s̄) ≤ k to 𝒫;
27 return s*;
```

limit of the solver is set to one to the next iteration. This is done to promote diversification to the search since taking the first integer solution of the solver is likely to produce new solutions far enough from the incumbent. Later, the last local branching constraint is removed (line 16), $\bar{s}$ is updated (line 17) and a new branching constraint is added to $\mathscr{P}$ (line 18).

TIMELIMIT : The execution of the model has been interrupted because it reached the given $timeLimitIt$. In such a case, if the objective function of the solution found is better than the overall best (line 20) the best solution is updated (lines 21 and 22) and the last local branching constraint is removed; otherwise, parameter k is decreased. Later, solution $\bar{s}$ is updated and a new branching constraint is added (lines 25 and 26).

4.4 Hybrid Solver

The proposed matheuristics are also explored in this thesis as an extra refinement phase for the previously described metaheuristics. Therefore, this Hybrid Solver (HS) is composed of three sequential steps: (*i*) the KHE solver is employed to generate a reasonable initial solution; (*ii*) the metaheuristic algorithm is employed to improve this solution as much as possible until stagnation, and; (*iii*) the matheuristic method is employed to provide fine improvement of the current solution until a timeout condition is reached. The matheuristic could be any of those proposed in this thesis; however, only Defect-Oriented Fix-and-Optimize (DO-FixOpt) was explored in this hybrid setup. Original Fix-and-Optimize algorithm was not explored in this setup because its extended version (DO-FixOpt) outperforms it, and Local Branching was also not selected due to its poor performance (see Table 5.12). The metaheuristic is considered stagnated when it reaches one twentieth of the available time without obtaining any improvement. The SVNS algorithm was chosen to compose the metaheutristic component of the hybrid solver due to its simplicity and superior performance when compared

with the other metaheuristics (see Table 5.10). A basic scheme of the hybrid solver, which will be abbreviated either as SVNS-DO-FixOpt or HS throughout this thesis, can be seen in Figure 4.3.

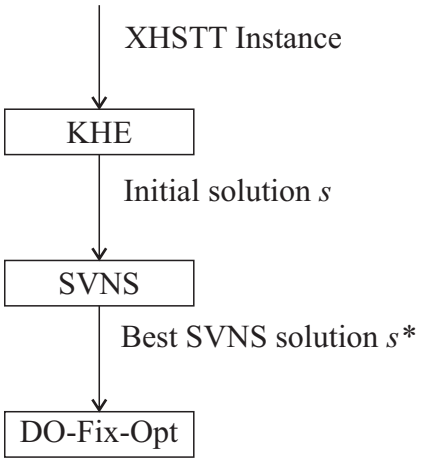


Figure 4.3: Scheme of the proposed Hybrid Solver of metaheuristic and matheuristic.

Chapter 5

Computational Experiments

This chapter presents the results obtained by the formulations and algorithms developed in this thesis. Section 5.1 presents the computational environment in which the experiments were performed. Section 5.2 describes the two instance sets considered on computational experiments: XHSTT-ITC2011-hidden and XHSTT-2014. Section 5.3 presents the results of the mathematical programming formulations and techniques proposed in this work. Section 5.4 presents the results of the heuristic algorithms developed in this thesis, grouped by technique: (*i*) VNS based, (*ii*) LAHC based, and (*iii*) matheuristics. A discussion is conducted at the end of the presentation of each result. Section 5.5 compares the best performing algorithms of each type and places these results according to the state-of-art methods in the literature. Finally, Section 5.6 presents the new lower bounds and best known solutions obtained in this thesis.

The results are expressed as a pair (H, S), in which H and S denote the cost of violating hard and soft constraints, respectively. When hard constraints are not violated, only the cost of soft constraint violation is presented. The solvers implemented in thesis, along with the solutions and reports obtained, can be found at the GOAL-UFOP website[1].

[1] `http://www.goal.ufop.br/software/hstt/`

5.1 Computational Environment

Computational experiments were performed on an Intel ® i7 4510U 2.6 Ghz PC with 8GB of RAM under Ubuntu 12.04 operating system. The software was coded in C++ and compiled using GCC 4.6.1. Results were validated by HSEval validator[2]. The academic version of the state-of-the-art solver Gurobi 6.5.1, with default parameter settings, was used to solve the mathematical programming models.

5.2 Instance Characterization

Two instance sets were considered to evaluate the algorithm and formulation performances: (*i*) ITC2011 Hidden Instances, and; (*ii*) XHSTT-2014 Instances. Both sets cover features of timetabling from several countries and have variate sizes, ranging from instances demanding the assignment of 75 lectures to instances that demand the assignment of thousands of lectures and resources. These sets are described in the next subsections.

5.2.1 ITC2011 Hidden Instances

ITC2011 Hidden Instances[3] (XHSTT-ITC2011-hidden) was considered because it was employed in the second phase of ITC2011. This phase dedicated to compare the algorithms in the same computational environment and runtime. Moreover, this instance set is currently the most used to evaluate algorithms for XHSTT in the literature. Table 5.1 presents the main features of this instance set. An abbreviation on the instance names was adopted to improve the readability. The abbreviation follows the pattern XX-YY-ZZ where XX represents the country that the instance is from, YY identifies the education institution and ZZ represents the year of the timetabling problem. Country abbreviations are: AU - Australia, BR - Brazil, DK - Denmark, FI - Finland, ES -

[2] `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi`
[3] `https://www.utwente.nl/ctit/hstt/archives/XHSTT-ITC2011-hidden/`

Spain, GR - Greece, IT - Italy, KS - Kosovo, NL - Netherlands, UK - United Kingdom, US - United States of America and ZA - South Africa. There is an overlap of instances in archives XHSTT-ITC2011-hidden and XHSTT-2014. Whenever the instance is the same, the abbreviation will be identical.

Table 5.1: Features of XHSTT-ITC2011-hidden archive and adopted abbreviations.

Abbrev.	Instance	Times	Resources	Events	Duration
BR-SA-00	BrazilInstance2	25	20	63	150
BR-DF-89	BrazilInstance3	25	24	69	200
BR-SM-00	BrazilInstance4	25	35	127	300
BR-SN-00	BrazilInstance6	25	44	140	350
FI-ES-12	FinlandElementarySchool	35	103	291	445
FI-SS-06	FinlandSecondarySchool2	40	79	469	566
GR-HS-11	AigioFirstHighSchool10-11	35	245	283	532
IT-I4-96	Italy_Instance4	36	99	748	1101
KS-PR-11	KosovaInstance1	62	164	809	1912
NL-KP-03	Kottenpark2003	38	587	1156	1203
NL-KP-05	Kottenpark2005	37	644	1235	1272
NL-KP-08	Kottenpark2008	40	126	1047	1118
NL-KP-09	Kottenpark2009	38	194	1166	1301
ZA-WD-09	Woodlands2009	42	70	278	1353
ES-SS-08	SpanishSchool	35	91	225	439
GR-P3-08	WesternGreeceUniversity3	35	25	210	210
GR-PA-08	WesternGreeceUniversity4	35	31	262	262
GR-P5-08	WesternGreeceUniversity5	35	24	184	184

Table 5.2 presents the occurrence of constraints and whether they are hard or soft in each instance. A hard constraint is denoted as H and a soft constraint is identified as S. An empty cell means that the constraint is not present in that instance.

Table 5.2: Constraints in XHSTT-ITC2011-hidden instances.

Instance	Assign Times	Assign Resources	Prefer Times	Prefer Resources	Link Events	Order Events	Spread Events	Avoid Split Assignments	Distribute Split Events	Split Events	Avoid Clashes	Avoid Unavailable Times	Limit Workload	Limit Idle Times	Limit Busy Times	Cluster Busy Times
BR-SA-00	H		S				H		S	H	H	H		S		S
BR-DF-89	H		S				H		S	H	H	H		S		S
BR-SM-00	H		S				H		S	H	H	H		S		S
BR-SN-00	H		S				H		S	H	H	H		S		S
FI-ES-12	H		H				S			H	H	H			S	
FI-SS-06	H						H			H	H			S	S	
GR-HS-11	H		S		H		H			H	H	H		S	S	
IT-I4-96	H		H				H			H	H	H		S	S	
KS-PR-11	H		H				H		S	H	H	H		H	H	
NL-KP-03	H	H	H	H	H		H			H	H	H		S	S	S
NL-KP-05	H	H	H	H	H		H			H	H	H		S	S	S
NL-KP-08	H	H	H	H	H		H			H	H	H		H		S
NL-KP-09	H	H	H	H	H		H			H	H	H		S		S
ZA-WD-09	H		S		S						H	S				
ES-SS-08	H	H		H			H			H	H	H			S	S
GR-P3-08	H						S				H	H		H	H	
GR-PA-08	H				H		S				H	H		H	H	
GR-P5-08	H						S				H	H		H	H	

5.2.2 XHSTT-2014 Instances

The other instances considered are those that compose the XHSTT-2014 set[4]. It contains some of the XHSTT-ITC2011-hidden but includes several new instances, from various countries, such as Australia, Denmark, and United States. This instance set was also considered because it is the newest one and several researchers are using it to record best known solutions and lower bounds. Table 5.3 presents the features of this instance set.

Table 5.3: Features of XHSTT-2014 archive

Instance	Times	Resources	Events	Duration
AU-BG-98	40	131	387	1564
AU-SA-96	60	99	296	1876
AU-TE-99	30	76	308	806
BR-SA-00	25	20	63	150
BR-SM-00	25	35	127	300
BR-SN-00	25	44	140	350
DK-FG-12	50	438	1077	1077
DK-HG-12	50	694	1235	1235
DK-VG-09	60	262	918	918
ES-SS-08	35	91	225	439
FI-PB-98	40	111	387	854
FI-WP-06	35	41	172	297
FI-MP-06	35	64	280	306
GR-H1-97	35	95	372	372
GR-P3-10	35	114	178	340
GR-PA-08	35	31	262	262
IT-I4-96	36	99	748	1101
KS-PR-11	62	164	809	1912
NL-KP-03	38	587	1156	1203
NL-KP-05	37	644	1235	1272
NL-KP-09	38	194	1148	1274
UK-SP-06	25	202	1227	1227
US-WS-09	100	242	628	6354
ZA-LW-09	148	37	185	838
ZA-WD-09	42	70	278	1353

Table 5.4 details the constraints considered in each instance.

[4] `https://www.utwente.nl/ctit/hstt/archives/XHSTT-2014/`

Table 5.4: Presence of constraints in XHSTT-2014 instances

Instance	Assign Times	Assign Resources	Prefer Times	Prefer Resources	Link Events	Order Events	Spread Events	Avoid Split Assignments	Distribute Split Events	Split Events	Avoid Clashes	Avoid Unavailable Times	Limit Workload	Limit Idle Times	Limit Busy Times	Cluster Busy Times
AU-BG-98	H	H	H	H	H		S	H	H	H	H	H	H			
AU-SA-96	H	H	H	H	H		S	S	H	H	H	H	H		S	
AU-TE-99	H	H		H	H		S	S	H	H	H	H	H		S	
BR-SA-00	H		H				H		S	H	H	H		S		S
BR-SM-00	H		H				H		S	H	H	H		S		S
BR-SN-00	H		H				H		S	H	H	H		S		S
DK-FG-12	H	H		H			S				H	H		S	S	S
DK-HG-12	H	H		H			S				H			S	S	H
DK-VG-09	H	H		H			S				H			S	S	S
ES-SS-08	H	H		H			H			H	H	H			S	S
FI-PB-98	H		H				H			H	H	H		S	S	
FI-WP-06	H		H				H			H	H			S	S	
FI-MP-06	H		H				H			H	H	H		S	S	
GR-HI-97	H				H		H				H	H				
GR-P3-10	H		S		H		H			H	H	H		S	S	
GR-PA-08	H				H		S				H	H		S	H	
IT-I4-96	H		H				H			H	H	H		S	S	S
KS-PR-11	H		H				H		S	H	H	H		H	H	
NL-KP-03	H	H	H	H	H		H			H	H	H		S	S	S
NL-KP-05	H	H	H	H	H		H			H	H	H		S	S	S
NL-KP-09	H	H	H	H	H		H			H	H	H		S		S
UK-SP-06	H	H			H		H				H			S		
US-WS-09	H	H	S	S			S			H	H					
ZA-LW-09	H		H		S		S				H					
ZA-WD-09	H		S		S		H				H	S				

5.3 Formulation Results

This section presents the results of the proposed mathematical programming formulations and techniques presented in the Chapter 3. Initially a comparison between the original formulation $\mathscr{F}_1$ and the proposed alternative formulation $\mathscr{F}_2$ is performed. In sequence, the effectiveness of the proposed Dantzig-Wolfe Column Generation and Cut-and-Solve approaches are evaluated.

5.3.1 Comparison between $\mathscr{F}_1$ and $\mathscr{F}_2$

The formulations $\mathscr{F}_1$ and $\mathscr{F}_2$ are compared in this subsection. First, the effectiveness of each proposed cut is evaluated. Further, the dimensions of the resulting mathematical programming models are compared. At last, the results of the linear relaxation and of the integer programming model for both $\mathscr{F}_1$ and $\mathscr{F}_2$ are compared.

Cut Effectiveness

Table 5.5 presents, for each cut, whether its addition to $\mathscr{F}_1$ changes the optimal solution of the linear relaxation of the model or not (i.e. whether isolated cuts of this type were found). Each entry in the table represents the execution of the linear relaxation of $\mathscr{F}_1$ applying the refereed cut on it. The cuts were ordered from the one that was effective to most instances to the one that was effective to least instances. The following abbreviations were used: Number of Busy Times Cut (NBT), Link X and Y Cut (LXY), Multi-commodity Flow (MCF), Link Y and Q Cut (LYQ), and Cluster Busy Times Cut (CBT).

From the analysis of Table 5.5, it can be concluded that NBT and LXY were the most effective cuts, improving the linear relaxation of all instances considered. NBT affects variables $q_{r,t}$, which are essential to calculate resource-related constraint penalties. LXY plays a central role since variables $y_{se,t}$ define the time assignment of the events and it provides a stronger link within the main variable, $x_{se,t,er,r}$. MCF does not affect

Table 5.5: Effectiveness of each cut over XHSTT-ITC2011-hidden archive.

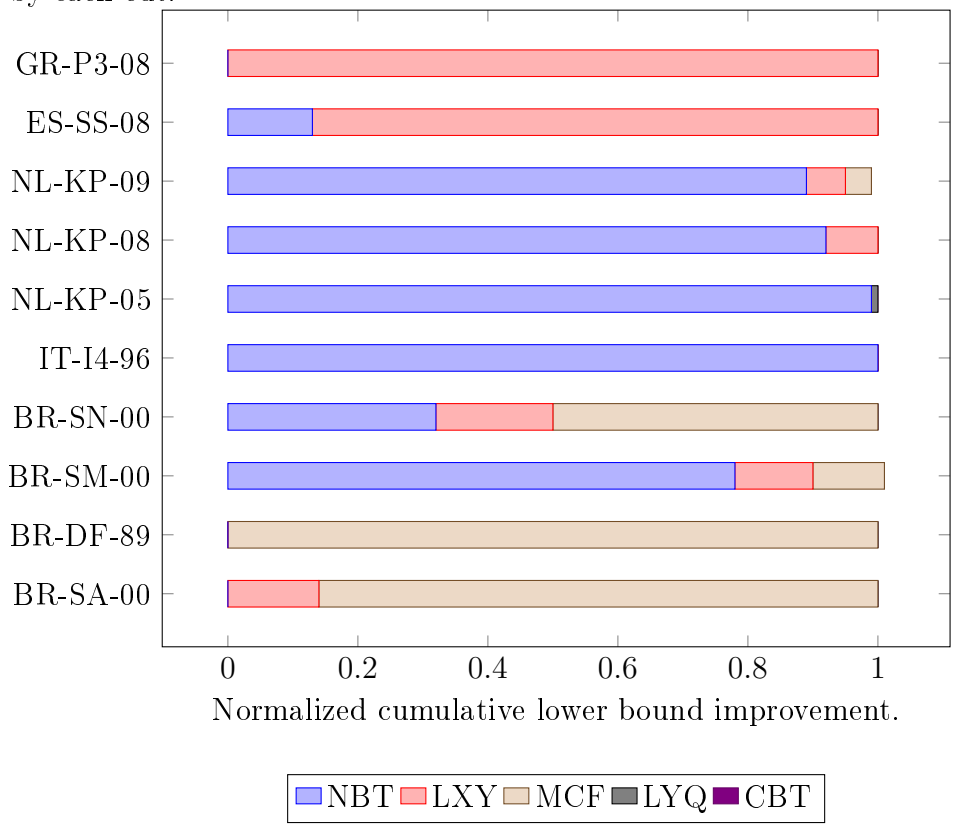
Instance	NBT	LXY	MCF	LYQ	CBT
BR-SA-00	✓	✓	✓		✓
BR-DF-89	✓	✓	✓		✓
BR-SM-00	✓	✓	✓		✓
BR-SN-00	✓	✓	✓		✓
FI-ES-12	✓	✓			
FI-SS-06	✓	✓	✓		
GR-HS-11	✓	✓	✓	✓	
IT-I4-96	✓	✓	✓	✓	
KS-PR-11	✓	✓	✓		
NL-KP-03	✓	✓	✓	✓	✓
NL-KP-05	✓	✓	✓	✓	✓
NL-KP-08	✓	✓	✓	✓	✓
NL-KP-09	✓	✓	✓	✓	✓
ZA-WD-09	✓	✓		✓	
ES-SS-08	✓	✓		✓	
GR-P3-08	✓	✓	✓		
GR-PA-08	✓	✓	✓	✓	
GR-P5-08	✓	✓	✓		

instances FI-ES-12, ZA-WD-09, and ES-SS-08. Indeed, MCF strengthens the definition of Limit Idle Times constraints and this constraint does not apply to these instances. LYQ is effective when the assignment and the preference of resources apply to the instances. It was also effective to strengthen the link events specification. As expected, CBT only affects the instances that this constraint applies to. Finally, it is possible to note that the features of the instances provide valuable information for selecting the cuts to be active. Therefore the activation of cuts could be tailored according to the features of the input problem.

Figure 5.1 presents the magnitude of lower bound improvement in the linear relaxation achieved by each cut. The following steps were executed to generate this data: (*i*) the linear relaxation of the original formulation is executed for each instance; (*ii*) the cuts were introduced, one by one, in a cumulative way and the partially improved lower bound values were recorded; (*iii*) these values were normalized in such a way that the lower bound provided by the original formulation is set as 0 and the lower bound

provided when all the proposed cuts are active is 1. Instances in which the optimal lower bound is 0 or no improvement is achieved by the alternative formulation were not included in the chart.

Figure 5.1: Normalized cumulative lower bound improvement in the linear relaxation achieved by each cut.



In Figure 5.1, it is possible to observe how much each cut contributes to the lower bound improvement achieved by the alternative formulation $\mathscr{F}_2$. NBT cuts increase the lower bound in most of the instances, specially for the Dutch and the Italian ones. LXY cuts were mostly effective in the Greek and Spanish instances, while MCF cuts were mostly effective in the Brazilian ones. Although LYQ and CBT virtually did not improve the lower bound, they were still considered on the alternative formulation because they cut fractional solutions of the search space as could be seen in Table 5.5. Since this lower bound improvement is cumulative, it is important to highlight that

these values of improvement would have been different if another cut inclusion order were adopted.

Model Dimensions

Figures 5.2, 5.3, and 5.4 represent, respectively, the number of variables, constraints, and non-zeros of formulations $\mathscr{F}_1$ and $\mathscr{F}_2$ applied to the instances of XHSTT-ITC2011-hidden archive.

Figure 5.2: Comparison of the number of variables of $\mathscr{F}_1$ and $\mathscr{F}_2$ for each instance in XHSTT 2011 hidden.

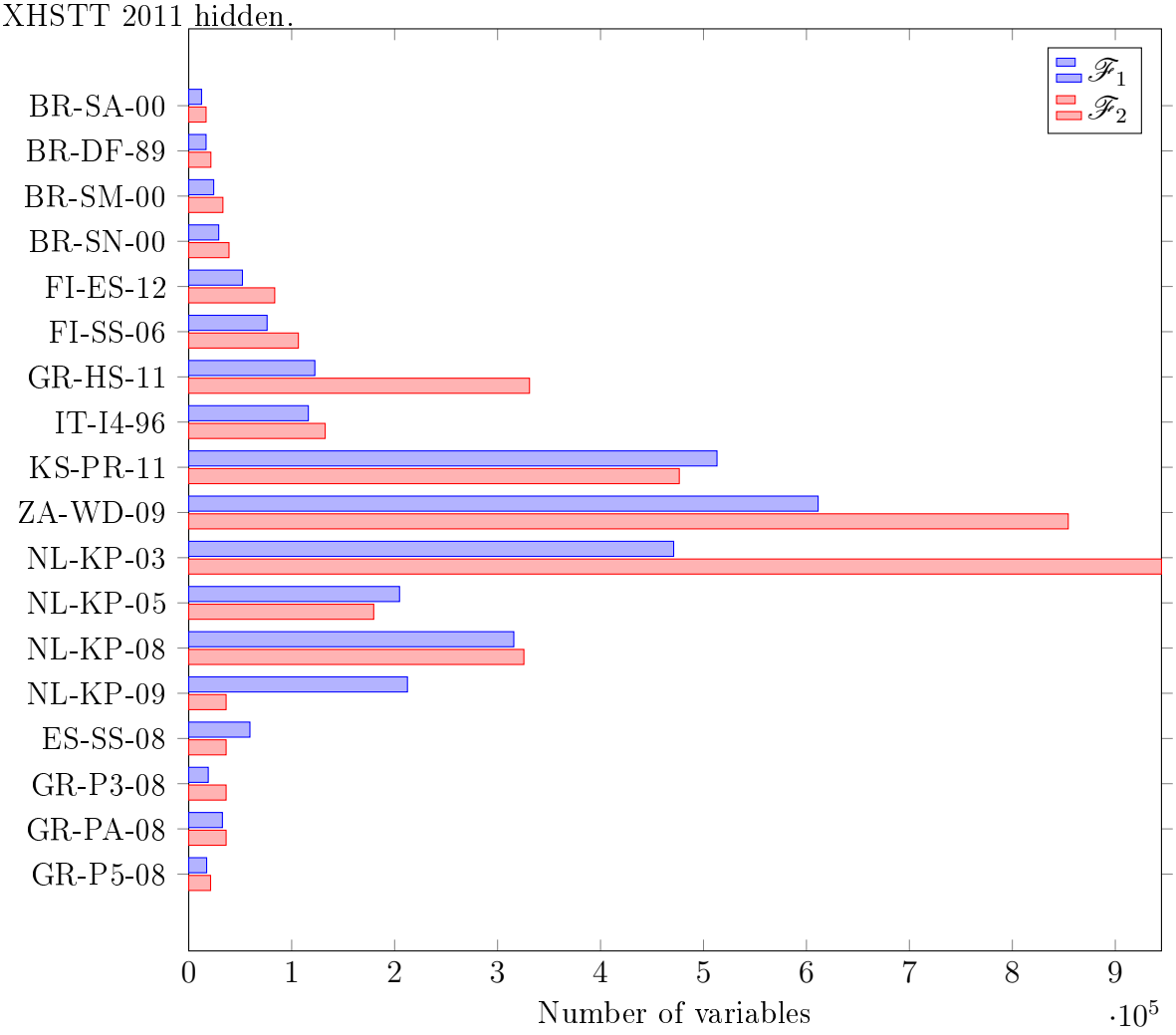
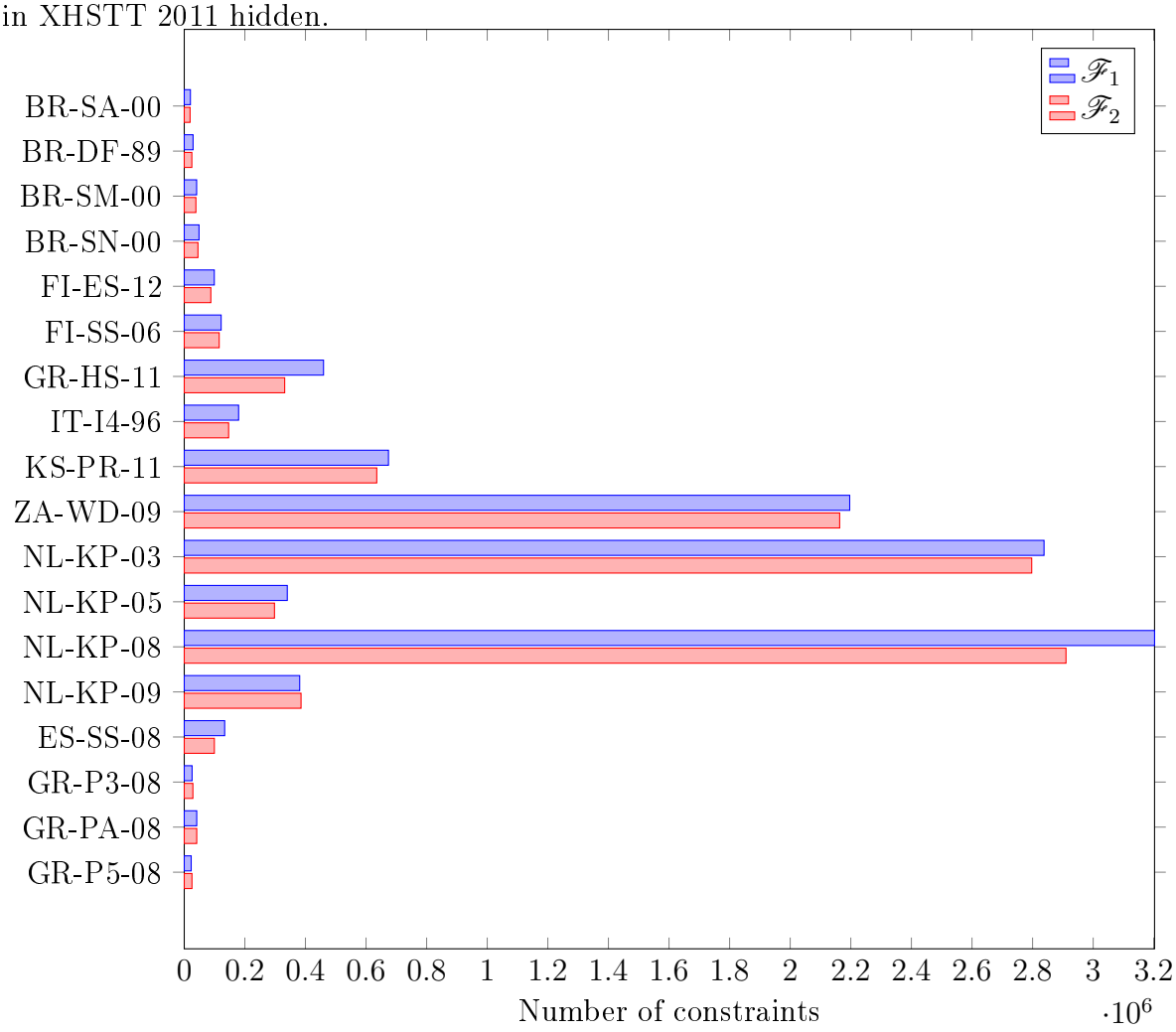
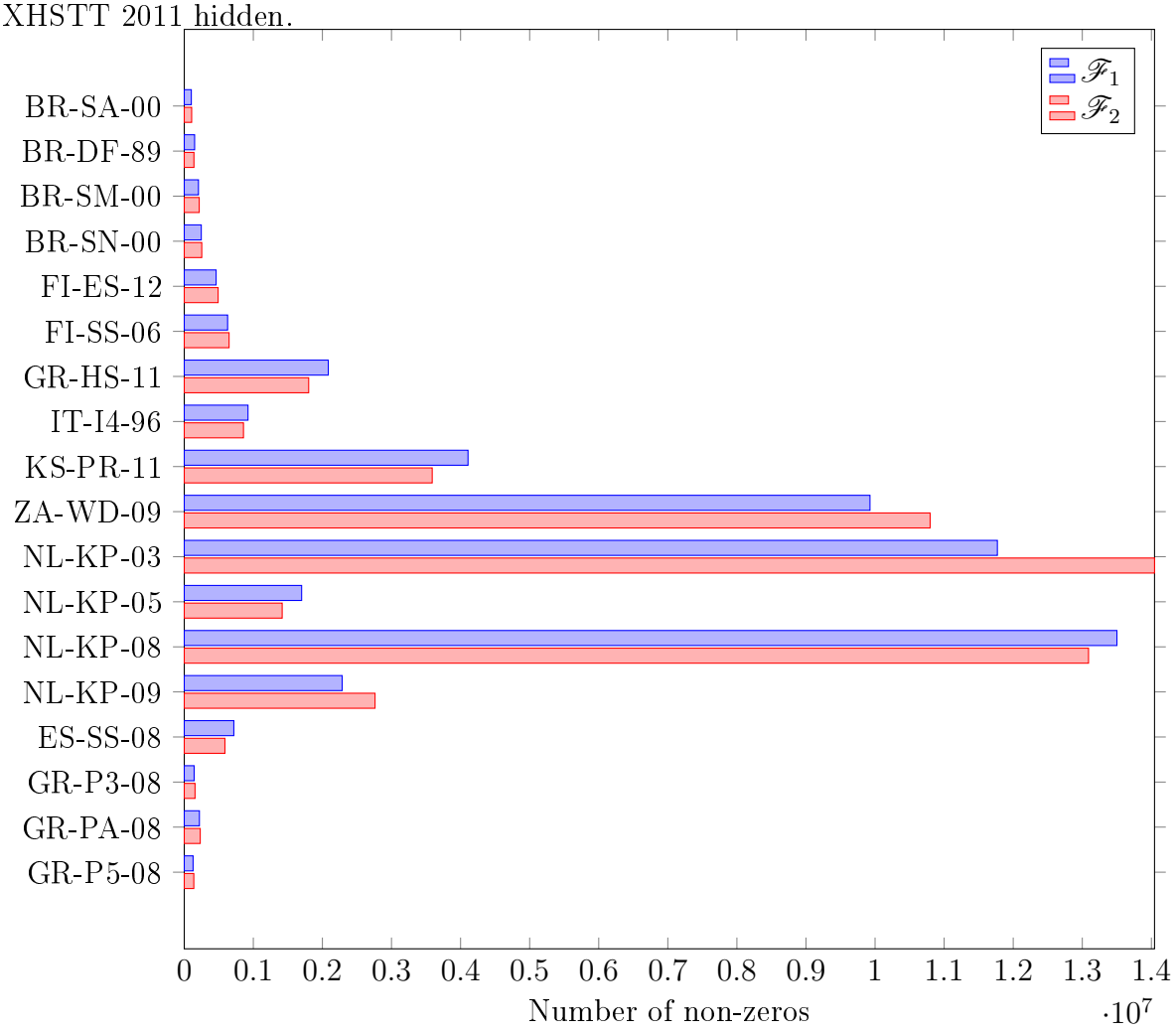


Figure 5.3: Comparison of the number of constraints of $\mathscr{F}_1$ and $\mathscr{F}_2$ for each instance in XHSTT 2011 hidden.



The number of variables in $\mathscr{F}_2$ is slightly higher than in $\mathscr{F}_1$ in most of the cases. This is mainly due to the new variables introduced to represent some of the arcs in the multicomodity flow reformulation. For some instances the increase rate is higher because they have a high number of resources (e.g. ZA-WD-09 and NL-KP-03). This reformulation forces the model to create a lot of graphs and therefore new variables. An odd behaviour can be observed for KS-PR-11, NL-KP-05, NL-KP-09, and ES-SS-08, in which the number of variables reduces in $\mathscr{F}_2$. This happens because a large number of variables to represent sub-events of infeasible duration is not generated in $\mathscr{F}_2$. In

Figure 5.4: Comparison of the number of non-zeros of $\mathscr{F}_1$ and $\mathscr{F}_2$ for each instance in XHSTT 2011 hidden.



such cases the reduction on the number of variables for infeasible sub-events surpasses the number of new variables generated. Recall that variables $v_{r,t}$ ($|\mathcal{R}| \times |\mathcal{T}|$) are also discarded in $\mathscr{F}_2$.

Although the cuts demand new constraints to the models, the number of constraints is slightly smaller in $\mathscr{F}_2$ compared to $\mathscr{F}_1$ for most cases. It happens because of the reformulations, whereas several constraints can be dropped from the $\mathscr{F}_1$. One example is Inequality (2.38), that generates a large number of constraints ($O(|\mathcal{R}| \times |\mathcal{TG}| \times |\mathcal{T}|^3)$) but can be removed when considering the multicomodity flow reformulation in $\mathscr{F}_2$.

Linear Relaxations

Table 5.6 presents linear relaxation of both original ($\mathscr{F}_1$) and alternative ($\mathscr{F}_2$) formulations over the instances of XHSTT-2011 and their running times as well. Column ↓Gap presents the percentage of decrease of the gap from using $\mathscr{F}_1$ to using $\mathscr{F}_2$ ($\frac{\mathcal{LB}_{\mathscr{F}_2} \times 100}{\mathcal{UB}} - \frac{\mathcal{LB}_{\mathscr{F}_1} \times 100}{\mathcal{UB}}$). The table also presents the best known solution for these instances ($\mathcal{UB}$). Optimal best known solutions are marked with an asterisk. Instances whose optimal solution cost is zero were not included. Lower bounds that were improved by the alternative formulation $\mathscr{F}_2$ are highlighted in bold.

Table 5.6: Comparison between linear relaxations of $\mathscr{F}_1$ and $\mathscr{F}_2$.

		Time (s)		$\mathcal{LB}$		
Instance	$\mathcal{UB}$	$\mathscr{F}_1$	$\mathscr{F}_2$	$\mathscr{F}_1$	$\mathscr{F}_2$	↓Gap
BR-SA-00	5*	0.2	0.4	0.5	**4.0**	70%
BR-DF-89	24*	0.2	0.4	0.0	**2.0**	8%
BR-SM-00	51*	0.5	1.1	8.0	**32.5**	48%
BR-SN-00	35*	2.5	4.4	2.0	**16.0**	40%
FI-ES-12	3*	6.1	10.9	0.0	0.0	0%
IT-I4-96	27*	15.8	25.6	0.0	**15.0**	56%
NL-KP-03	420	28,147.0	10,320.1	0.0	0.0	0%
NL-KP-05	784	31,756.3	17,879.5	0.0	**86.0**	11%
NL-KP-08	15463	35.5	35.1	2.3	**2904.0**	19%
NL-KP-09	5095	3,115.6	8,751.9	0.1	**179.0**	4%
ES-SS-08	335	24.4	23.4	54.9	**305.0**	75%
GR-P3-08	5*	1.8	2.3	2.0	**5.0**	60%
GR-PA-08	3*	6.5	3.4	0.0	0.0	0%
Average						30%

Table 5.6 shows that the linear relaxations provided by $\mathscr{F}_2$ are significantly stronger that the ones from $\mathscr{F}_1$, leading to an average gap reduction of 30%. The processing times were slightly longer in $\mathscr{F}_2$ in most cases. For NL-KP-09 in special, the new formulation was considerably slower (2.8×), but it conducted to a huge linear relaxation improvement. Such a lower bound was even better than the previously best known for this instance in the integer model (see Table 5.15). The lower bound of 2904 for NL-KP-08 was also better than the previously best know bound. It should also be noticed

that $\mathscr{F}_2$ was considerably faster than $\mathscr{F}_1$ for the most time demanding instances, NL-KP-03 and NL-KP-05. This is due to the high number of resources in these instances, where the multicomodity flow reformulation seems to speed up the linear relaxation optimization process.

Integer Model

Table 5.7 presents the integer results for both original and alternative formulations under one hour of time limit. Instances in which neither of the formulations achieved a feasible solution within the given time limit were omitted. Optimal bounds are marked with an asterisk and the best bounds/Gaps are highlighted in bold.

Table 5.7: Integer Programming results for formulations $\mathscr{F}_1$ and $\mathscr{F}_2$ within 1 hour time limit.

	$\mathscr{F}_1$			$\mathscr{F}_2$		
Instance	$\mathcal{LB}$	$\mathcal{UB}$	Gap	$\mathcal{LB}$	$\mathcal{UB}$	Gap
BR-SA-00	4.5	10	0.55	**5.0***	**5***	**0.00**
BR-DF-89	21.3	24	0.11	**24.0***	**24***	**0.00**
BR-SM-00	49.7	138	0.64	**51.0***	**51***	**0.00**
BR-SN-00	17.5	224	0.92	**35.0***	**217**	**0.84**
FI-ES-12	**3.0***	**3***	**0.00**	2.7	4	0.33
GR-HS-11	0.0*	0*	0.00	0.0*	0*	0.00
IT-I4-96	27.0*	**11244**	1.00	27.0*	15348	1.00
GR-P3-08	5.0*	9	0.44	5.0*	**6**	**0.17**
GR-PA-08	0.0	24	1.00	**2.0**	**8**	**0.75**
GR-P5-08	0.0*	0*	0.00	0.0*	0*	0.00
Average			0.47			**0.31**

From the analysis of Table 5.7, it is possible to note that $\mathscr{F}_2$ found the optimal lower bound for eight out of ten instances against five optimal lower bounds in $\mathscr{F}_1$. Regarding the upper bounds the alternative formulation was also superior, finding five optimal bounds against three from the original formulation. Overall, the Gaps produced by $\mathscr{F}_2$ were considerably smaller than the ones provided by $\mathscr{F}_1$, being, on average, 16% smaller.

Although some progress towards finding better bounds for timetabling through mathematical programming was done in this work, neither of the formulations were able to find any feasible solution for the eight remaining instances in XHSTT-ITC2011-hidden. This result shows that this problem is still challenging for the research community, since exact methods are still not able to handle a considerable set of instances of this problem.

5.3.2 Column Generation Results

Table 5.8 presents the results of the Dantzig-Wolfe Column Generation approach developed in this work. Column Iters presents the number of iterations required to solve the problem, column Cols. presents the number of variables required to solve the problem, column $\mathcal{LB}$ presents the lower bound obtained, and column Time (s) the running time. Results of the linear relaxation of $\mathscr{F}_2$ are also reproduced in the table for comparison purposes. Similarly to Table 5.6, column ↓Gap presents the percentage of decrease of the gap from using $\mathscr{F}_1$ to using $\mathscr{F}_2$ ($\frac{\mathcal{LB}_{\mathscr{F}_2} \times 100}{\mathcal{UB}} - \frac{\mathcal{LB}_{\mathscr{F}_1} \times 100}{\mathcal{UB}}$). Lower bounds marked with '*' are optimal and bold values highlight the bounds that were improved by Dantzig-Wolfe Column Generation.

As it can be noticed from Table 5.8, the Dantzig-Wolfe Column Generation was able to tighten the lower bound of the linear relaxation for five instances in the XHSTT-ITC2011-hidden set. Two of these lower bounds match the optimal solution ($\mathcal{UB}$). On average, the proposed Column Generation approach led to an 18% Gap reduction when compared with the alternative formulation $\mathscr{F}_2$. However, it is important to highlight that the processing time of Dantzig-Wolfe Column Generation was considerably higher than the processing time of $\mathscr{F}_2$. Indeed, Dantzig-Wolfe Column Generation could not even terminate for the Dutch instances within 24 hours of processing time.

Although some bounds were improved by Dantzig-Wolfe Column Generation, the incorporation of this decomposition in a Branch-and-Price [7] framework was not explored in this thesis. The high processing time required to solve the Dantzig-Wolfe

Table 5.8: Results of the proposed Column Generation approach for educational timetabling.

		$\mathscr{F}_2$		$\mathscr{CG}$				
Instance	$\mathcal{UB}$	Time (s)	$\mathcal{LB}$	Iters	Cols.	Time (s)	$\mathcal{LB}$	↓Gap
BR-SA-00	5*	0.4	4.0	25	285	19.5	**5.0***	20%
BR-DF-89	24*	0.4	2.0	28	322	27.2	2.0	0%
BR-SM-00	51*	1.1	32.5	58	314	116.1	**45.0**	25%
BR-SN-00	35*	4.4	16.0	17	380	51.6	**29.0**	37%
FI-ES-12	3*	10.9	0.0	32	2272	729.0	**1.0**	33%
IT-I4-96	27*	25.6	15.0	58	848	1,005.7	**27.0***	44%
NL-KP-03	420	10,320.1	0.0	-	-	24h+	-	-
NL-KP-05	784	17,879.5	86.0	-	-	24h+	-	-
NL-KP-09	15463	35.1	2904.0	-	-	24h+	-	-
NL-KP-09	5059	8,751.9	179.0	-	-	24h+	-	-
ES-SS-08	335	23.4	305.0	145	5295	7,481.6	305	0%
GR-P3-08	5*	2.3	5.0*	18	327	20.6	5.0*	0%
GR-PA-08	3*	3.4	0.0	16	219	33.1	0.0	0%
Average								18%

Column Generation and the high number of columns generated make it impractical to solve an extended Branch-and-Price model to obtain integer solutions. Such an approach would be applicable only to very small instances, whose optimal solution can already be found through integer programming or heuristic algorithms in a very short processing time.

5.3.3 Cut-and-Solve Results

Table 5.9 presents the results of the Cut-and-Solve method under one hour of time limit. Columns $\mathcal{LB}$, $\mathcal{UB}$ and, Gap present, respectively, the lower bound, the upper bound, and the gap between them for each instance. Results of the linear relaxation of $\mathscr{F}_2$ are also reproduced in the table for comparison purposes. Bounds marked with '*' are optimal and bold values highlight better performance when comparing the solution approaches. As suggested by Climer and Zhang [18], it was assumed $\alpha = 0.1$ to perform the experiments.

Table 5.9: Comparison between Cut-and-Solve approach and Integer Programming formulation $\mathscr{F}_2$ within 1 hour time limit.

	$\mathscr{F}_2$			$\mathscr{C}\mathscr{S}$			
Instance	$\mathcal{LB}$	$\mathcal{UB}$	Gap	Iters	$\mathcal{LB}$	$\mathcal{UB}$	Gap
BR-SA-00	5.0*	5*	0.00	1	5.0*	5*	0.00
BR-DF-89	**24.0***	24*	**0.00**	1	20.0	24*	0.17
BR-SM-00	51.0*	**51***	**0.00**	1-	51.0	67	0.24
BR-SN-00	**35.0***	217	**0.84**	1-	17.0	**157**	0.89
FI-ES-12	**2.7**	4	**0.33**	1	1.0	**3***	0.67
GR-HS-11	0.0*	**0***	**0.00**	1-	0.0*	-	1.00
IT-I4-96	**27.0***	**15348**	1.00	1-	0.0	-	1.00
GR-P3-08	5.0*	6	0.17	1	5.0*	**5***	**0.00**
GR-PA-08	**2.0**	8	0.75	1-	1.9	**4**	**0.52**
GR-P5-08	0.0*	0*	0.00	1	0.0*	0*	0.00
Average			**0.31**				0.51

As can be seen by Table 5.9, the Cut-and-Solve approach was not effective for this model of educational timetabling. In column Iters, one can note that, for all the instances, Cut-and-Solve either required one iteration to be solved or could not terminate within one hour of time limit. Indeed, for educational timetabling instances, the number of decision variables having reduced cost other than zero is quite small. Therefore, solving the first iteration of the Cut-and-Solve algorithm is virtually the same as solving the standard Integer Programming model.

To the best of the author's knowledge, Cut-and-Solve was only tested on Travelling Salesman [18] and Facility Location problems [79]. In these category of problems, the decision variables often have a reduced cost other than zero - it is directly related to the distance (or cost) of the arc related to the decision variable. Therefore, due to the features of the problem, Cut-and-Solve seems to be unsuitable to handle educational timetabling problems.

5.4 Algorithm Results

This section presents the results of the developed algorithms presented in Chapter 4. Subsections 5.4.1, 5.4.2 and 5.4.3 present, respectively, the results of Variable Neighbourhood Search, Late Acceptance Hill Climbing, and Matheuristic based algorithms. These results are compared with the initial solution to evaluate how effective each algorithm was at improving it.

In this section, the experiments were executed following the ITC2011 rules: the time limit was adjusted to be equivalent to 1,000 seconds in the benchmark provided by the organizers (870 seconds) and the number of available threads was set to 1. Note that the use of commercial mathematical programming solvers, such as Gurobi, were not allowed in the ITC2011. However, it is considered in the experiments because the objective is to determinate which algorithmic strategy is the best to tackle the problem rather than determinate which one would have won the competition.

The algorithms were also compared according to ranking procedure of ITC2011: each algorithm was executed 5 times in each instance, and the average result was recorded. Each solver received a ranking on each instance, from 1 (best) to N (worst), according to the average costs obtained (N is the number of algorithms compared). Solvers with smaller average rank are claimed better.

5.4.1 VNS Results

Table 5.10 presents the results obtained by the VNS method and its variants. Initial solutions provided by KHE14 are also present in the table for comparison purposes. The average objective function cost of the five executions for each instance are shown in each respective cell. The value of "Ranking" was calculated following the ITC2011 ranking rules. The best results are highlighted in bold.

In some instances, even the production of feasible solutions is challenging, specially when most constraints are set as hard ones. VNS method and its variants were able to

Table 5.10: Results of VNS and variants.

Instance	KHE14	RVNS	SVND	VNS	SVNS
BR-SA-00	44.0	44.0	42.2	33.8	**29.0**
BR-DF-89	109.0	109.0	109.0	106.6	**104.8**
BR-SM-00	(12.0, 128.0)	(12.0, 119.0)	(9.6, 111.8)	(6.4, 113.0)	**(6.4, 110.6)**
BR-SN-00	145.0	145.0	145.0	123.4	**121.0**
FI-ES-12	3.0	3.0	3.0	3.0	3.0
FI-SS-06	18.0	18.0	8.0	**0.0**	**0.0**
GR-HS-11	10.0	10.0	9.6	**0.8**	**0.8**
IT-I4-96	54.0	54.0	53.8	50.6	**50.2**
KS-PR-11	20.0	20.0	20.0	**13.0**	13.8
NL-KP-03	1515.0	1419.4	**1329.6**	1474.8	1384.4
NL-KP-05	(19.2, 5758.2)	(15.4, 6144.0)	(15.4, 6549.2)	(18, 5399.0)	**(13.8, 6812.4)**
NL-KP-08	(25.0, 26861.0)	(19.4, 28299.4)	(19.8, 34548.4)	(15.6, 28114.8)	**(15.4, 31039.4)**
NL-KP-09	(16.0, 7930.0)	(13.4, 28781.0)	(12.4, 26085.0)	**(11.0, 12793.0)**	(11.0, 12995.0)
ZA-WD-09	(26.0, 0.0)	(25.0, 0.0)	(23.4, 0.0)	**(6.8, 0.0)**	(7.0, 0.0)
ES-SS-08	1117.0	1117.0	1112.0	**959.6**	**959.6**
GR-P3-08	10.0	10.0	10.0	**5.0**	**5.0**
GR-PA-08	16.0	16.0	16.0	**5.0**	5.2
GR-P5-08	0.0	0.0	0.0	0.0	0.0
Ranking	4.36	3.81	3.22	2.00	**1.61**

improve significantly the feasibility penalty for the 5 instances in which KHE14 could not provide a feasible initial solution.

As shown in Table 5.10, the SVNS algorithm presented the best results among the VNS variants. An explanation to this result is the fact that SVNS can accept worse solutions along the iterations, which makes it more suitable to escape from local minima. The RVNS algorithm presented poor performance. Since it does not use a local search method, it loses performance by trying large neighbourhoods instead of smaller ones. Such large neighbourhoods slow down the search for good solutions.

5.4.2 LAHC Results

Table 5.11 presents the average objective function values obtained by LAHC and its variants. Similarly to the previous section, initial solutions provided by KHE14 are also present in the table and the average rankings are reported.

An analysis of the results of Table 5.11 indicates that the LAHC algorithm and its variants improved considerably the initial solutions provided by KHE14 for most of the

Table 5.11: Results of LAHC and variants.

Instance	KHE14	SA	LAHC	SF-LAHC	SA-SF-LAHC
BR-SA-00	44.0	44.0	34.4	33.8	**32.0**
BR-DF-89	109.0	**108.4**	109.0	109.0	109.0
BR-SM-00	(12.0, 128.0)	**(4.8, 156.8)**	(8.0, 116.6)	(7.6, 111.2)	(5.6, 139.4)
BR-SN-00	145.0	145.0	130.0	**112.6**	114.4
FI-ES-12	3.0	3.0	3.0	3.0	3.0
FI-SS-06	18.0	18.0	**0.0**	**0.0**	**0.0**
GR-HS-11	10.0	10.0	10.0	10.0	10.0
IT-I4-96	54.0	54.0	40.8	**37.4**	39.0
KS-PR-11	20.0	20.0	8.0	8.6	**7.6**
NL-KP-03	1515.0	1345.2	1528.4	**1290.6**	1390.6
NL-KP-05	(19.2, 5758.2)	(15.2, 9924.8)	(14.8, 7670.0)	(14.8, 6730.4)	**(14.4, 9090.0)**
NL-KP-08	(25.0, 26861.0)	(25.0, 26861.0)	(25.0, 26861.0)	(25.0, 26861.0)	(25.0, 26861.0)
NL-KP-09	(16.0, 7930.0)	(16.0, 7930.0)	(16.0, 7930.0)	(16.0, 7930.0)	(16.0, 7930.0)
ZA-WD-09	(26.0, 0.0)	(8.0, 0.0)	**(6.2, 0.0)**	(7.2, 0.0)	**(6.2, 0.0)**
ES-SS-08	1117.0	1117.0	954.0	960.0	**929.0**
GR-P3-08	10.0	10.0	5.2	**5.0**	**5.0**
GR-PA-08	16.0	16.0	5.8	**4.4**	4.8
GR-P5-08	0.0	0.0	0.0	0.0	0.0
Ranking	4.06	3.47	2.94	2.33	**2.14**

instances in the XHSTT-ITC2011-hidden archive. LAHC was considerably better than SA. This is an interesting result, since LAHC is a new metaheuristic and it is still open for improvements. In addition, the Simulated Annealing is known as a good algorithm for dealing with scheduling problems.

The Stagnation Free version of LAHC obtained promising results, outperforming its original version in several instances. Indeed, SF-LAHC is more suited than its original version to escape from local optima. Finally, it is worthy to highlight the strong performance observed for the combination of LAHC and SA, proposed in this work (SA-SF-LAHC).

5.4.3 Matheuristic Results

Table 5.12 presents the average results obtained with the matheuristic based algorithms proposed in this work. Again, KHE14 initial solutions and the average rankings are also presented for comparison purposes.

Table 5.12: Results of matheurisitc based algorithms.

Instance	KHE14	LB	Fix-Opt	DO-FixOpt	SVNS-DO-FixOpt
BR-SA-00	44.0	16.0	**5.0**	**5.0**	**5.0**
BR-DF-89	109.0	98.0	27.2	27.6	**26.6**
BR-SM-00	(12.0, 128.0)	122.0	69.0	68.4	**65.8**
BR-SN-00	145.0	145.0	53.4	45.4	**43.6**
FI-ES-12	3.0	3.0	3.0	3.0	3.0
FI-SS-06	18.0	18.0	**0.0**	**0.0**	**0.0**
GR-HS-11	10.0	10.0	**0.0**	**0.0**	**0.0**
IT-I4-96	54.0	52.0	27.8	**27.0**	**27.0**
KS-PR-11	20.0	20.0	7.8	**3.4**	4.6
NL-KP-03	1515.0	1515.0	1399.6	**1297.2**	1421.8
NL-KP-05	(19.2, 5758.2)	(22.0, 5139.0)	**(16.8, 7381.6)**	(17.0, 6817.0)	(18.6, 7249.8)
NL-KP-08	(25.0, 26861.0)	(25.0, 26861.0)	(17.8, 117602.4)	(14.8, 125873.2)	**(14.2, 105092.6)**
NL-KP-09	(16.0, 7930.0)	(16.0, 7930.0)	(16.2, 26435.0)	(16.4, 25717.0)	**(12.8, 14136.0)**
ZA-WD-09	(26.0, 0.0)	(26.0, 0.0)	25.8	17.8	**14.4**
ES-SS-08	1117.0	1112.0	655.8	**493.0**	494.0
GR-P3-08	10.0	10.0	**5.0**	**5.0**	**5.0**
GR-PA-08	16.0	15.0	5.0	**4.6**	**4.6**
GR-P5-08	0.0	0.0	0.0	0.0	0.0
Ranking	4.44	4.22	2.61	2.00	**1.72**

The proposed matheuristic was able to find feasible solutions for two instances in which KHE14 and the metaheuristics could not: BR-SM-00 and ZA-WD-09. However, even the matheuristics could not reach feasible solutions on most of Dutch instances. Indeed, these instances are very challenging and, to the best of the author's knowledge, no solver could find feasible solutions for them within the ITC2011 time limit and rules.

Defect-Oriented Fix-and-Optimize presented remarkable results, beating by far its original version. The selection of resources based on solution defects and shared events allowed the DO-FixOpt algorithm to find good solutions faster than its original version. DO-FixOpt and SVNS-DO-FixOpt were able to consistently find the optimal solution for seven instances (namely BR-SA-00, FI-ES-12, FI-SS-06, GR-HS-11, IT-I4-96, GR-P3-08, and GR-P5-08), even within this very short time-limit and one thread setup.

SVNS Defect-Oriented Fix-and-Optimize presented the best results among the matheuristic based algorithms. However the DO-FixOpt presented a very similar performance. This can be explained by the fact of applying SVNS in the beginning of the search helps the algorithm to find good solutions faster before proceeding to the math-

euristic phase. This situation is more evident when evaluating the performance on large instances, in which the time-limit is too short for the matheuristic to be effective. In this case SVNS ensures that a good solution will be found. On the other hand, for small and easy instances, the time that SVNS consumes might slow down the search for good solutions. A feature that should be considered is that SVNS-DO-FixOpt is more robust than DO-FixOpt. SVNS-DO-FixOpt can perform well even on huge sized instances whereas standalone DO-FixOpt could not perform well within the same time limit.

Although Local Branching could find several improved solutions regarding KHE14 initial ones, it was not competitive with the problem specific Fix-and-Optimized approaches proposed in this work.

By evaluating these results, it should be noticed that problem specific information is very important to choose the variables to release in the matheuristic approaches for timetabling. Indeed, this is a factor of differentiation of top performing approaches and average solvers.

5.5 Overall Comparison of Solvers

Table 5.13 reproduces the results of the top performing solver of each different category explored in this thesis. In this table, the dominance of the Fix-and-Optimize based approach (SVNS-DO-FixOpt) is clear: it achieved a ranking considerably better than the second ranked solver (1.81 against 2.56). Metaheuristic based approaches presented a very similar performance, despite SVNS had a slightly better average performance compared to SA-SF-LAHC. The pure Integer Programming approach using the alternative formulation ($\mathscr{F}_2$) had a good performance for the small instances. However it requires more processing time than the ITC2011 time limit to produce any competitive solution for the others.

Table 5.13: Comparison of different solution techniques proposed in this work.

Instance	KHE14	IP $\mathscr{F}_2$	SA-SF-LAHC	SVNS	SVNS-DOFixOpt
BR-SA-00	44.0	**5.0**	32.0	29.0	**5.0**
BR-DF-89	109.0	**24.0**	109.0	104.8	26.6
BR-SM-00	(12.0, 128.0)	**51.0**	(5.6, 139.4)	(6.4, 110.6)	65.8
BR-SN-00	145.0	217.0	114.4	121.0	**43.6**
FI-ES-12	**3.0**	4.0	**3.0**	**3.0**	**3.0**
FI-SS-06	18.0	-	**0.0**	**0.0**	**0.0**
GR-HS-11	10.0	**0.0**	10.0	0.8	**0.0**
IT-I4-96	54.0	15348.0	39.0	50.2	**27.0**
KS-PR-11	20.0	-	7.6	13.8	**4.6**
NL-KP-03	1515.0	-	1390.6	**1384.4**	1421.8
NL-KP-05	(19.2, 5758.2)	-	(14.4, 9090.0)	**(13.8, 6812.4)**	(18.6, 7249.8)
NL-KP-08	(25.0, 26861.0)	-	(25.0, 26861.0)	(15.4, 31039.4)	**(14.2, 105092.6)**
NL-KP-09	(16.0, 7930.0)	-	(16.0, 7930.0)	**(11.0, 12995.0)**	(12.8, 14136.0)
ZA-WD-09	(26.0, 0.0)	-	(6.2, 0.0)	(7.0, 0.0)	**14.4**
ES-SS-08	1117.0	-	929.0	959.6	**494.0**
GR-P3-08	10.0	6.0	**5.0**	**5.0**	5.2
GR-PA-08	16.0	8.0	4.8	5.2	**4.6**
GR-P5-08	0.0	0.0	0.0	0.0	0.0
Ranking	4.14	3.78	2.72	2.56	**1.81**

Figure 5.5 presents a comparison of convergence for the best performing solvers of each category presented in this thesis (IP $\mathscr{F}_2$, SA-SF-LAHC, SVNS, and SVNS-DO-FixOpt) on instance BR-SA-00. This instance was selected to compose the chart because it captures the main features about the convergence of the algorithms. The optimal solution for this instance has a cost of 5 unities and it is represented by the dashed line at the bottom of the chart.

From the analysis of Figure 5.5, it can be seen that the metaheuristic algorithms cannot escape from a local optima achieved by around 300 seconds of processing time and they spend most of the processing time stuck on it. The Fix-and-Optimize based algorithm can easily overcome this limitation and reached the optimal solution rather quickly. The IP solver also reached the optimal solution for this instance. However it took more processing time when compared to the Fix-and-Optimize based solver. A similar behaviour about the algorithms can be observed for the other instances, except regarding IP $\mathscr{F}_2$, that often cannot provide any feasible solution to the instances within 870 seconds.

Figure 5.5: Convergence chart of SVNS-DO-FixOpt, SVNS, SA-SF-LAHC, and IP $\mathcal{F}_2$ for instance BR-SA-00.

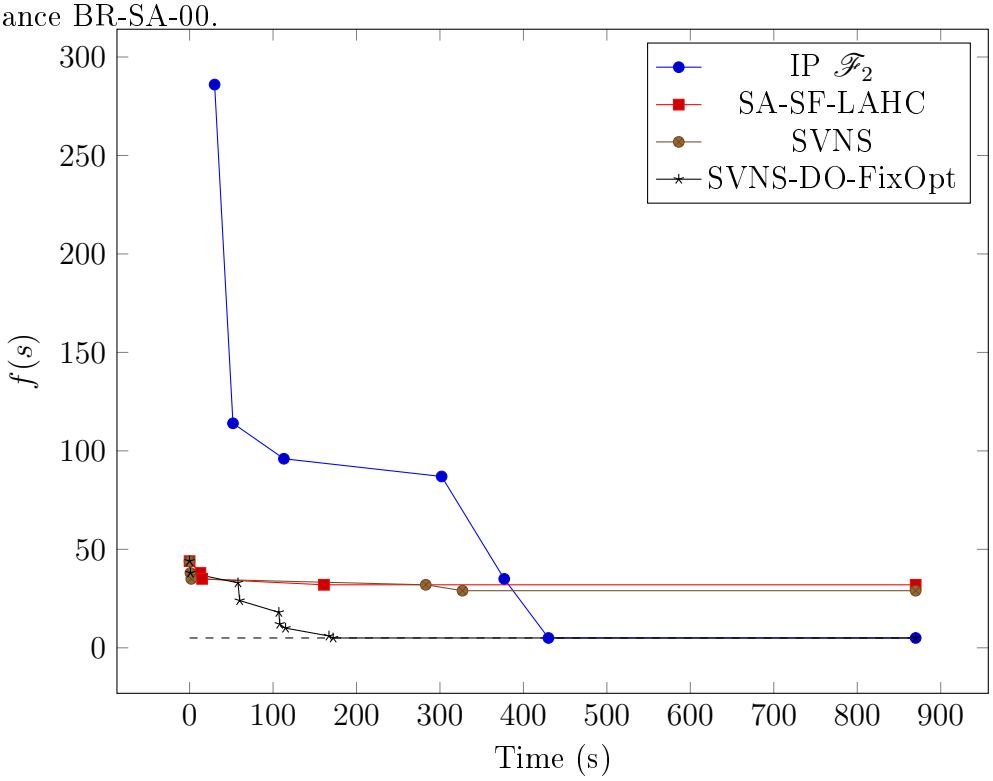


Table 5.14 presents a comparison of the best performing algorithm developed in this thesis, SVNS-DO-FixOpt, with state-of-art approaches for educational timetabling. For a short description of these solvers, please refer to Section 1.2.

In Table 5.14, it is clear the superiority of SVNS-DO-FixOpt solver over the others: from 18 instances in the benchmark set, it achieved the best solution for 17 of them, implying on a remarkable ranking of 1.19. Besides the rankings, the objective function of the solutions provided by SVNS-DO-FixOpt are considerably better than the other solvers'. For example, look at instances BR-SA-00, GR-HS-11, and IT-I4-96, whose solutions achieved by SVNS-DO-FixOpt are optimal.

Furthermore, SVNS-DO-FixOpt could consistently find feasible solutions for 15 out of 18 solutions of the XHSTT-ITC2011-hidden archive within the ITC2011 time limit. Meanwhile, KHE and MaxSAT achieved feasible solutions for 13 instances, RPGD achieved for 12, and GOAL for 9 instances.

Table 5.14: Comparison of SVNS-DO-FixOpt with state-of-art approaches for educational timetabling.

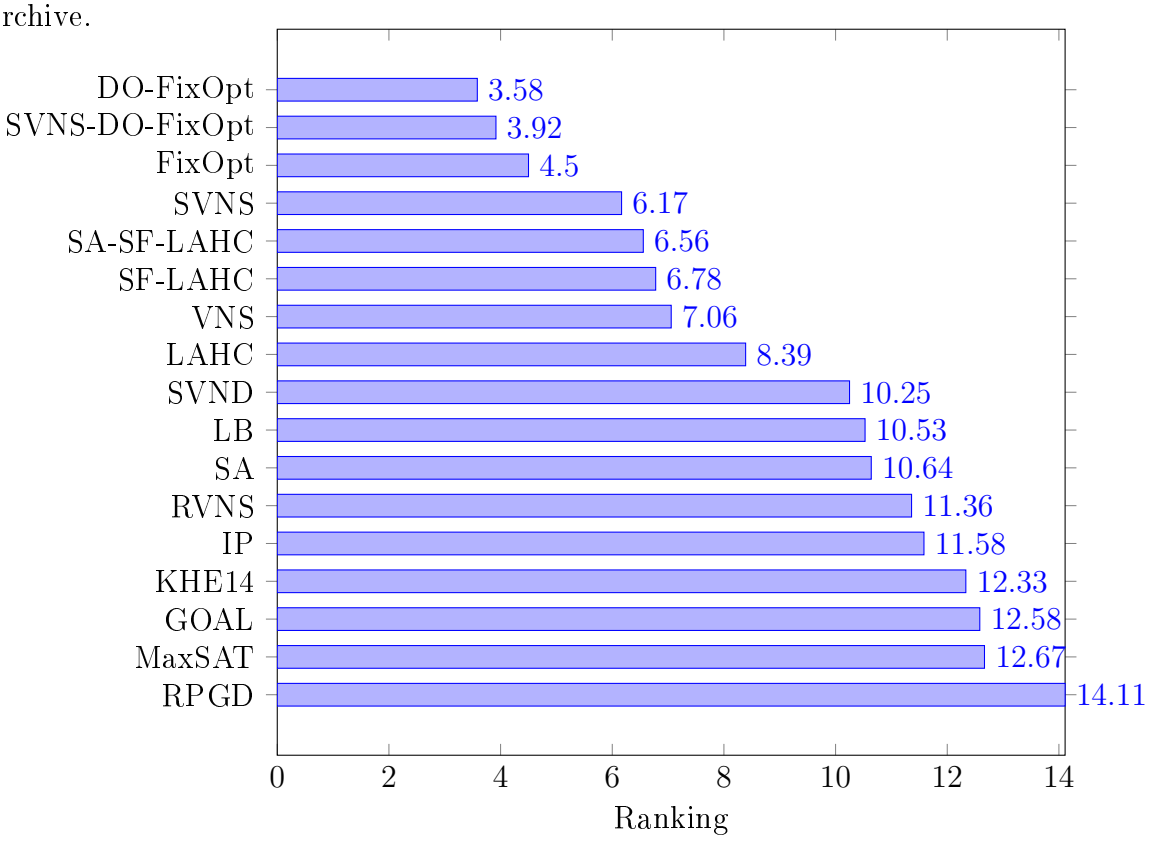
Instance	KHE14[43]	MaxSAT[24]	RPGD[3]	GOAL[35]	SVNS-DOFixOpt
BR-SA-00	44.0	57.0	96.0	(1.0, 63.9)	**5.0**
BR-DF-89	109.0	75.0	152.0	127.8	**26.6**
BR-SM-00	(12.0, 128.0)	214.0	(10.0, 143.0)	(17.2, 99.6)	**65.8**
BR-SN-00	145.0	352.0	266.0	(4.0, 233.5)	**43.6**
FI-ES-12	**3.0**	**3.0**	4.0	4.0	**3.0**
FI-SS-06	18.0	3523.0	9.0	0.4	**0.0**
GR-HS-11	10.0	4582.0	596.0	15.3	**0.0**
IT-I4-96	54.0	16979.0	520.0	658.4	**27.0**
KS-PR-11	20.0	29946.0	(17.0, 9084.0)	(14.0, 6934.4)	**4.6**
NL-KP-03	1515.0	-	40862.0	(0.6, 90195.8)	**1421.8**
NL-KP-05	(19.2, 5758.2)	-	(26.0, 26129.0)	(33.9, 27480.4)	**(18.6, 7249.8)**
NL-KP-08	(25.0, 26861.0)	-	(24.0, 999999.0)	(25.7, 31403.7)	**(14.2, 105092.6)**
NL-KP-09	(16.0, 7930.0)	-	(22.0, 999999.0)	(36.6, 154998.5)	**(12.8, 14136.0)**
ZA-WD-09	(26.0, 0.0)	**0.0**	(2.0, 279.0)	(2.0, 15.8)	14.4
ES-SS-08	1117.0	-	1451.0	865.2	**494.0**
GR-P3-08	10.0	7.0	10.0	5.6	**5.2**
GR-PA-08	16.0	141.0	19.0	7.4	**4.6**
GR-P5-08	**0.0**	**0.0**	2.0	**0.0**	**0.0**
Ranking	2.83	3.64	3.69	3.58	**1.19**

Figure 5.6 presents the ITC rankings of all the solvers proposed in this thesis together with the best performing state-of-art approaches.

From the analysis of Figure 4.2 it can be seen that the Fix-and-Optimize based algorithms (DO-FixOpt, SVNS-DO-FixOpt, and FixOpt) completely dominated the ranking as the best solvers for educational timetabling. Although SVNS-DO-FixOpt achieved a better ranking in Table 5.12 , DO-FixOpt got a slightly better ranking than SVNS-DO-FixOpt in this joint analysis. This could be explained by the fact that the introduction of other solvers may affect the rankings between two solvers: a small difference on the average solutions of an instance can lead to a large difference on the rankings if several other solvers achieve solutions in between of these two values. Neglecting the magnitude of the difference between the solutions provided by different solvers is, indeed, a flaw in the ITC2011 ranking method.

On a second tier of performance comes most of the metaheuristic based approaches developed in this work, namely SVNS, SA-SF-LAHC, SF-LAHC, VNS, and LAHC. Indeed, they use the same neighbourhood moves and are quite similar to each other. The

Figure 5.6: Overall comparison of rankings of solvers over the XHSTT-ITC2011-hidden archive.



main differentiation factor here is the method of escaping from local optima, whereas the ones employed in SVNS and in Stagnation-Free LAHC seems to be the best.

The remaining solvers come in the last tier. Despite the overall performance of them was inferior, some features should be highlighted. First of all, although KHE14 achieved a bad ranking, it came before all the other methods in the state-of-art literature. Indeed, all the algorithms presented in this thesis, except for the IP solver, use KHE14 initial solutions as an initial point. Consequently they are expected to introduce improvements on these solutions and to perform better than KHE14. Improvements on KHE14 can affect the performance of all solvers proposed in this thesis. Therefore, it is something to be explored in future research on the topic. For example, GOAL solver, winner of ITC2011, uses an old version of KHE to generate initial solutions. The improvements

introduced in this new version of KHE were so remarkable that it even slightly overcame the GOAL solver.

Although the IP solver achieved a bad ranking, an interesting behaviour can be noted about its performance: it achieved optimal solutions for five of the instances in the evaluated instance set, but it could not produce any competitive solution for most of the others. This result indicates that, even solvers having a bad ranking could be the most appropriate for a specific set of instances.

5.6 Improving Best Known Bounds

The 25 instances of the XHSTT-2014 archive were used to estimate the efficiency of the best performing algorithm in this thesis (SVNS-DO-FixOpt) when larger processing times are available. SVNS-DO-FixOpt will be here refereed to as Hybrid Solver (HS). In these tests, the Hybrid Solver was allowed to run for 36,000 seconds (10 hours) and two combinations were considered:

BKS-HS: Hybrid Solver is applied considering the current best known solution as the initial solution.

KHE-HS: Hybrid Solver is applied using the KHE14 to provide initial solutions, such as the previous experiments.

On the one hand, the intention behind testing the BKS-HS is to evaluate the refinement capacity of the proposed algorithm. This setup is particularly useful if good solutions are available, such as the scheduling of the last semester, for example. It is important to remind that it was one of the phases of the ITC2011 competition. On the other hand, the KHE-HS is more general since it applies to most real world cases, in which initial solutions for the considered problem are not known a priori.

Regarding the lower bounds, the alternative integer programming formulation $\mathscr{F}_2$ also ran for 36,000 seconds on the open instances of XHSTT-2014. The results obtained by these two combinations and by the alternative formulation $\mathscr{F}_2$ are shown in Table 5.15. Additionally, the previously best known lower bounds ($\mathcal{LB}$) and solutions ($\mathcal{UB}$) before this work are also reproduced in the table. For BKS-HS, results better than the current best known solution are highlighted in bold. For KHE-HS, whenever a result is better than or equals to the current best know it is highlighted. New lower bounds obtained are also highlighted in bold. In BKS-HS column, results marked with a dash mean that they are already optimal and it does not make sense to try to load these solutions and to improve them. In IP $\mathscr{F}_2$ column, lower bounds marked with a dash mean they are already optimal. Optimal bounds are marked with an asterisk.

The proposed hybrid solver obtained remarkable results on polishing the best known solutions: 15 out of 16 non-optimal solutions of the instance set considered were improved. New optimal solutions were found for instances AU-SA-96, AU-TE-99, FI-MP-06, FI-WP-06, GR-PA-08, and IT-I4-96. Taking into account the experiments in which the KHE14 was used to produce initial solutions, one can conclude that the results obtained are strong: the optimal solution has been found in several instances and, for some, the result obtained was even better than the previous best known solution. However, for some datasets, such as the Australian and the Dutch ones, the standalone solver still cannot achieve feasible solutions. Regarding the lower bounds, strong results were also obtained. Four new best known lower bounds were found. Among them, the new lower bound for AU-SA-96 allowed to claim optimality for the instance.

Finally, it should be noticed that the time spent on optimization (10 hours) is not critical in this application since the schedule can be performed weeks, or even months, before the beginning of the semester.

Table 5.15: New best known bounds obtained in this thesis for XHSTT-2014 archive.

	Prev. known		New $\mathcal{LB}$	New $\mathcal{UB}$	
Instance	$\mathcal{LB}$	$\mathcal{UB}$	IP $\mathcal{F}_2$	BKS-HS	KHE-HS
AU-BG-98	0	(1, 386)	0	**126**	(2, 398)
AU-SA-96	0	24	0*	**0***	(3, 21)
AU-TE-99	0	125	**20***	**20***	(1, 36)
BR-SA-00	5	5	-	-	**5***
BR-SM-00	51	51	-	-	**51***
BR-SN-00	35	35	-	-	**35***
DK-FG-12	285	3310	**412**	**1263**	**1668**
DK-HG-12	(7, 0)	(12, 3124)	(7, 0)	**(12, 2330)**	(12, 3371)
DK-VG-09	(0, 0)	(2, 4097)	**(2, 0)**	**(2, 2323)**	**(2, 2765)**
ES-SS-08	334	335	334	335	351
FI-PB-98	0	0	-	-	**0***
FI-WP-06	0	1	0*	**0***	2
FI-MP-06	77	83	77	**77***	**77***
GR-H1-97	0	0	-	-	**0***
GR-P3-10	0	0	-	-	**0***
GR-PA-08	3	4	3	**3***	**3***
IT-I4-96	27	34	27	**27***	**27***
KS-PR-11	0	0	-	-	**0***
NL-KP-03	0	617	0	**199**	1103
NL-KP-05	89	1078	89	**425**	(8, 4460)
NL-KP-09	170	9180	**180**	**1620**	(7, 64470)
UK-SP-06	0	(16, 2258)	0	**(5, 4014)**	(53, 1524)
US-WS-09	0	697	0	**103**	(124)
ZA-LW-09	0	0	-	-	**0***
ZA-WD-09	0	0	-	-	**0***

Chapter 6

Concluding Remarks

This chapter summarizes the main findings and contributions of this thesis to the scientific community of combinatorial optimization, in specific to the automated timetabling field. Section 6.1 presents the main conclusions that can be drawn from this thesis. Section 6.2 presents the contributions of this thesis in both fields of mathematical programming and algorithms for timetabling. Section 6.3 suggestions of future work on this topic are presented. At the end, in Section 6.4, a list of the papers produced during this thesis is presented.

6.1 Conclusions

The practical and theoretical aspects of educational timetabling attracted the interest of researchers in the last years. A good scheduling of activities in educational context may improve the staff satisfaction and the students' performance. New formulations and algorithms for XHSTT timetabling were explored in this thesis.

In the field of mathematical programming it can be concluded that, even with proposal of an enhanced alternative formulation for educational timetabling, this problem is still very challenging for the scientific community. Indeed, only five out of the eighteen instances on XHSTT-ITC-2011 dataset could be solved to optimality in a reasonable processing time. For eight of these instances not even a feasible solution was achieved within one hour of processing time. The proposed column generation approach showed

promising results, however it takes an infeasible amount of time to provide integer solutions to the unsolved instances.

Regarding the algorithmic approaches, Fix-and-Optimize based methods are consistently achieving the best results for this model of education timetabling problem. This result is surprising since it is strongly believed by the scientific community that metaheuristic approaches are the overall top performing algorithms for this category of problems.

Overall, this thesis provided solid grounds for further research on the topic. Besides of the proposal and/or implementation of several approaches for automated educational timetabling, it grouped together several other existing approaches in the literature and provided strong conclusions on which are the best solvers for this problem. Furthermore, this work was conducted on a standard widespread format and instance set, which could be easily followed be the scientific community.

6.2 Contributions

In the field of mathematical programming formulations, the main contributions of this thesis are: (i) the development of a stronger formulation $\mathscr{F}_2$ that incorporates new valid inequalities, pre-processing techniques and a multi-commodity network flow reformulation; and (ii) the development of a Dantzig-Wolfe column generation approach for XHSTT. These mathematical programming techniques developed in this work provided four new lower bounds out of twelve open instances in the XHSTT-2014 archive. The alternative formulation $\mathscr{F}_2$ was also better than the original one in the search for integer solutions. Cut-and-Solve algorithm was also explored in this thesis but, due to the problem features, it had poor performance.

In the field of algorithms, the main contributions of this thesis are: (i) the development of a problem specific Fix-and-Optimize approach for educational timetabling that is now the best performing approach for this problem by a large margin and pro-

duced new best known solutions for virtually all the open instances in the XHSTT-2014 archive; (*ii*) the design of an enhanced approach to select the variables to be freed in the Fix-and-Optimize algorithm, named Defect-Oriented FixOpt, whose performance overcame its original version and showed a huge potential to be explored in other problems besides the educational timetabling context; and (*iii*) the implementation and development of several metaheuristics for educational timetabling, in specific based on Variable Neighbourhood Search and Late Acceptance Hill-Climbing, and variations of these algorithms which overcame their original versions and showed potential to be explored in other combinatorial optimization problems.

6.3 Future Work

The following activities are suggested as topics of future work related to this thesis:

Develop a Graphical User Interface to handle XHSTT:

The XHSTT format is complex and virtually only timetabling researchers are able to work with this file format by now. A graphical user interface (GUI) to handle XHSTT would be essential to abstract the user from XHSTT knowledge and to make any person able to specify and solve timetabling problems through this format. Although some work has been done in this sense [32], there is still much room for improvement in the existing GUI for XHSTT.

Encode and solve new timetabling problems through XHSTT:

The XHSTT format is powerful enough to model most of the features of the educational timetabling problems, having sixteen different constraint types that might be hard, soft, or absent, besides allowing the generic specification of any type of resources and events. Although the encoding of some University Timetabling problems as XHSTT was explored in Fonseca *et al.* [33], there are still several problems that could be

modelled as XHSTT. The encoding of new timetabling problems as XHSTT and the solution of them through XHSTT solvers could be a large step towards the standardization of the instances, solvers and solutions for educational timetabling. Indeed, the lack of standardization is often mentioned as a major issue to the development of this research field [60, 62, 68].

Develop a new decomposition approach for timetabling:

Decomposition of large Integer Programming models is a promising technique to solve timetabling problems. Decomposition techniques consists in to decompose a large IP model into two or more sub-models, or stages. Each sub-model solves a particular task of the IP model. For example, the considered problem could be decomposed in two sub-problems: (*i*) assign times and split events, and; (*ii*) assign resources. This can reduce the number of variables and non-zeros, which can make the intractable model suitable for IP solvers[65]. This idea was successfully explored by Sorensen and Stidsen [70] to solve Danish timetabling problems. However, to the best of author's knowledge, it was never evaluated for XHSTT format and instances.

6.4 Publications

The following papers, produced during this thesis, were published in journals related to Operational Research:

- Fonseca, G. H., Santos, H. G., Carrano, E. G., and Stidsen, T. J. Integer programming techniques for educational timetabling. *European Journal of Operational Research*, pages –, 2017.
- Fonseca, G. H. G., Santos, H. G., and Carrano, E. G. Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, 74:108–117, 2016.

- Fonseca, G. H. G., Santos, H. G., and Carrano, E. G. Late acceptance hill-climbing for high school timetabling. *Journal of Scheduling*, 19(4):453–465, 2016.

The following papers were published and presented in international conferences related to scheduling and timetabling during this work:

- Fonseca, G. H. G., Santos, H. G., and Carrano, E. G., and Stidsen, T. J. R. Modelling and Solving University Course Timetabling Problems Through XHSTT. In *Proceedings of the 10th Conference on Practice and Theory of Automated Timetabling*, pages 127–138, 2016.

- Fonseca, G. H. G., Santos, H. G., and Carrano, E. G. Improving upper bounds in high school timetabling by matheuristics. In *Proceedings of the 7th Multidisciplinary International Conference on Scheduling: Theory and Applications*, pages 267–275, 2015.

Bibliography

[1] Abramson, D. Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Science*, 37(1):98–113, 1991.

[2] Abuhamdah, A. Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. *International Journal of Computer Science and Network Security*, 10(1):192–199, 2010.

[3] Ahmed, L. N., Özcan, E., and Kheiri, A. Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, 42(13):5463 – 5471, 2015.

[4] Asín Achá, R. and Nieuwenhuis, R. Curriculum-based course timetabling with sat and maxsat. *Annals of Operations Research*, 218(1):71–91, 2014.

[5] Avella, P. and Vasilev, I. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8(6):497–514, 2005.

[6] Badoni, R. P., Gupta, D., and Mishra, P. A new hybrid algorithm for university course timetabling problem using events based on groupings of students. *Computers & Industrial Engineering*, 78(1):12 – 25, 2014.

[7] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., and Vance, P. H. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[8] Bettinelli, A., Cacchiani, V., Roberti, R., and Toth, P. An overview of curriculum-based course timetabling. *Journal of the Spanish Society of Statistics and Operations Research*, 23(2):313–349, 2015.

[9] Birbas, T., Daskalaki, S., and Housos, E. Timetabling for greek high schools. *Journal of the Operational Research Society*, 48(12):1191–1200, 1997.

[10] Boschetti, M., Maniezzo, V., Roffilli, M., and Röhler, A. B. Matheuristics: Optimization, simulation and control. In Blesa, M. J., Blum, C., Di Gaspero, L., Roli, A., Sampels, M., and Schaerf, A., editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 171–177. Springer Berlin Heidelberg, 2009.

[11] Brito, S. S. and Santos, H. G. Automatic integer programming reformulation using variable neighborhood search. *Electronic Notes in Discrete Mathematics*, 58:7 – 14, 2017. 4th International Conference on Variable Neighborhood Search.

[12] Burke, E. K. and Bykov, Y. A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 2008.

[13] Burke, E. K. and Bykov, Y. The late acceptance hill-climbing heuristic. Technical Report CSM-192, Department of Computing Science and Mathematics, University of Stirling, June 2012.

[14] Burke, E. K. and Bykov, Y. The late acceptance hill-climbing heuristic. *European Journal of Operational Research*, 258(1):70 – 78, 2017.

[15] Burke, E. K. and Newall, J. P. A multistage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[16] Burke, E., Eckersley, A., McCollum, B., Petrovic, S., and Qu, R. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 206(1):46 – 53, 2010.

[17] Ceschia, S., Di Gaspero, L., and Schaerf, A. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, 39(7):1615–1624, 2012.

[18] Climer, S. and Zhang, W. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. *Artificial Intelligence*, 170(8):714 – 738, 2006.

[19] Conejo, A. J., Castillo, E., Minguez, R., and Garcia-Bertrand, R. *Decomposition techniques in mathematical programming: engineering and science applications.* Springer Berlin Heidelberg, 1 edition, 2010.

[20] Danna, E., Rothberg, E., and Le Pape, C. Integrating mixed integer programming and local search: A case study on job-shop scheduling problems. In *Fifth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR 2003)*, pages 65–79, 2003.

[21] Dantzig, G. B. and Wolfe, P. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[22] de Haan, P., Landman, R., Post, G., and Ruizenaar, H. A case study for timetabling in a Dutch secondary school. *Lecture notes in computer science: VI Practice and theory of automated timetabling. Berlin : Springer*, 3867:267–279, 2007.

[23] Demir, L., Tunalı, S., and Eliiyi, D. T. An adaptive tabu search approach for buffer allocation problem in unreliable non-homogenous production lines. *Computers & Operations Research*, 39(7):1477–1486, 2012.

[24] Demirović, E. and Musliu, N. Maxsat-based large neighborhood search for high school timetabling. *Computers & Operations Research*, 78:172 – 180, 2017.

[25] Di Gaspero, L. and Schaerf, A. Multi-neighbourhood local search with application to course timetabling. In *Proceedings of the 4th International Conference on the Practice and theory of automated timetabling*, pages 262–275. Springer, 2003.

[26] Domrös, J. and Homberger, J. An evolutionary algorithm for high school timetabling. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pages 485–488, 2012.

[27] Dorneles, A. P., Olinto, C. B. A., and Buriol, L. S. A column generation approach to high school timetabling modeled as a multicommodity flow problem. *European Journal of Operational Research*, 256(3):685 – 695, 2017.

[28] El-Sherbiny, M. M., Zeineldin, R. A., and El-Dhshan, A. M. Genetic algorithm for solving course timetable problems. *International Journal of Computer Applications*, 124(10), 2015.

[29] Even, S., Itai, A., and Shamir, A. On the complexity of timetable and multicommodity flow problems. *SIAM Jounal of Computing*, 5(4):691–703, December 1976.

[30] Fischetti, M. and Lodi, A. Local branching. *Mathematical Programming*, 98(1):23–47, 2003.

[31] Fischetti, M. and Lodi, A. Optimizing over the first chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

[32] Fonseca, G. H. G., Delfino, T. D., and Santos, H. G. A web-software to handle xhstt timetabling problems. In *Proceedings of the 9th Conference on Practice and Theory of Automated Timetabling*, pages 601–605, 2014.

[33] Fonseca, G. H. G., Santos, H. G., Carrano, E. G., and Stidsen, T. J. R. Modelling and Solving University Course Timetabling Problems Through XHSTT. In *Proceedings of the 11th International Conference on the Practice and Theory of Automated Timetabling*, pages 127–138, 2016.

[34] Fonseca, G. H. G., Santos, H. G., Toffolo, T. A. M., Brito, S. S., and Souza, M. J. F. A SA-ILS approach for the High School Timetabling Problem. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pages 493–495, 2012.

[35] Fonseca, G. H. G., Santos, H. G., Toffolo, T. A. M., Brito, S. S., and Souza, M. J. F. Goal solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, 239(1):77–97, 2016.

[36] Goodman, M. D., Dowsland, K. A., and Thompson, J. M. Hybridising grasp and network flows in the solution of a medical school scheduling problem. *Journal of Scheduling*, 15(6):717–731, 2012.

[37] Gotlieb, C. C. The construction of class-teacher time-tables. In *Proc. IFIP Congress*, pages 73–77. North Holland Pub. Co., 1963.

[38] Hansen, P. and Mladenovic, N. *Variable Neighborhood Search: A Chapter of Handbook of Applied Optimization.*, chapter 8. Les Cahiers du GERAD G-2000-3. Montreal, Canada, 2000.

[39] Hansen, P. and Mladenović, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[40] Kelley, J. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, pages 703–712, 1960.

[41] Kheiri, A., Özcan, E., and Parkes, A. J. HySST: Hyper-heuristic Search Strategies and Timetabling. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pages 497–499, 2012.

[42] Kingston, J. H. A tiling algorithm for high school timetabling. *Lecture notes in computer science: V Practice and theory of automated timetabling. Berlin: Springer*, 3616:208–225, 2005.

[43] Kingston, J. H. KHE14: An algorithm for high school timetabling. In *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling*, pages 26–29, 2014.

[44] Kingston, J. H. A Software Library for School Timetabling, available at http://sydney.edu.au/engineering/it/ jeff/khe/. Accessed in April, 2017.

[45] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[46] Kostuch, P. The university course timetabling problem with a three-phase approach. In *Proceedings of the 5th international conference on Practice and Theory of Automated Timetabling*, pages 109–125, Berlin, Heidelberg, 2005. Springer-Verlag.

[47] Kristiansen, S. and Stidsen, T. J. R. Elective course student sectioning at danish high schools. *Annals of Operations Research*, 239(1):99–117, 2016.

[48] Kristiansen, S., Sørensen, M., and Stidsen, T. R. Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, 18(4):377–392, 2015.

[49] Lewis, R., Paechter, B., and Mccollum, B. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Technical report, QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1. 0, Queen's University, Belfast, United Kingdom, 2007.

[50] Maniezzo, V., Stützle, T., and Voß, S. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer, 1st edition, 2010.

[51] McCollum, B., McCullam, P., Burke, E. K., Parkes, A. J., and Qu, R. The second international timetabling competition: Examination timetabling track. Technical report, QUB/IEEE/Tech/ITC2007/Exam/v4.0/17, Queen's University, Belfast, United Kingdom, 2007.

[52] Mladenovic, N. and Hansen, P. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[53] Moura, A. V. and Scaraficci, R. A. A grasp strategy for a more constrained school timetabling problem. *International Journal of Operational Research*, 7(2):152–170, 2010.

[54] Muller, T. ITC2007 solver description: a hybrid approach. *Annals OR*, 172(1):429–446, 2009.

[55] Neufeld, G. A. and Tartar, J. Graph coloring conditions for the existence of solutions to the timetable problem. *Communications of the ACM*, 17(8):450–453, August 1974.

[56] Nguyen, K., Nguyen, Q., Tran, H., Nguyen, P., and Tran, N. Variable neighborhood search for a real-world curriculum-based university timetabling problem. In *Third International Conference on Knowledge and Systems Engineering*, pages 157–162. IEEE, 2011.

[57] Nurmi, K. and Kyngas, J. A framework for school timetabling problem. In *Proceedings of the 3rd Multidisciplinary International Scheduling Conference: Theory and Applications*, pages 386–393, 2007.

[58] Ostermann, R. and de Werra, D. Some experiments with a timetabling system. *Operations-Research-Spektrum*, 3(4):199–204, 1982.

[59] Özcan, E., Bykov, Y., Birben, M., and Burke, E. K. Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of the 11th Congress on Evolutionary Computation*, pages 997–1004, Piscataway, NJ, USA, 2009.

[60] Pillay, N. A survey of school timetabling research. *Annals of Operations Research*, 218(1):261–293, 2014.

[61] Post, G., Di Gaspero, L., Kingston, J. H., McCollum, B., and Schaerf, A. The third international timetabling competition. *Annals of Operations Research*, 239(1):69–75, 2016.

[62] Post, G., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., and Schaerf, A. XHSTT: an XML archive for high school timetabling problems in different countries. *Annals of Operations Research*, 218(1):295–301, 2014.

[63] Puchinger, J. and Raidl, G. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer Berlin Heidelberg, 2005.

[64] Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., and Lee, S. Y. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, 12(1):55–89, 2009.

[65] Ruszczyński, A. Decomposition methods in stochastic programming. *Mathematical programming*, 79(1-3):333–353, 1997.

[66] Santos, H. G., Ochi, L. S., and Souza, M. J. F. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *ACM Journal of Experimental Algorithmics*, 10:2–9, 2005.

[67] Santos, H. G., Uchoa, E., Ochi, L. S., and Maculan, N. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, 2012.

[68] Schaerf, A. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999.

[69] Schmidt, G. and Ströhlein, T. Timetable construction - an annotated bibliography. *The Computer Journal*, 23(4):307–316, 1980.

[70] Sørensen, M. and Dahms, F. H. W. A two-stage decomposition of high school timetabling applied to cases in denmark. *Computers & Operations Research*, 43:36 – 49, 2014.

[71] Sørensen, M. and Stidsen, T. J. R. Hybridizing integer programming and metaheuristics for solving high school timetabling. *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling*, pages 557–560, 2014.

[72] Sørensen, M., Kristiansen, S., and Stidsen, T. R. International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, pages 489–492, 2012.

[73] Souza, M. J. F., Maculan, N., and Ochi, L. S. A grasp-tabu search algorithm for solving school timetabling problems. In *Metaheuristics: Computer decision-making*, pages 659–672. Springer, 2004.

[74] Tuga, M., Berretta, R., and Mendes, A. A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In *6th Annual IEEE/ACIS International Conference on Computer and Information Science*, pages 400–405. IEEE Computer Society, 2007.

[75] Valourix, C. and Housos, E. Constraint programming approach for school timetabling. *Computers & Operations Research*, pages 1555–1572, 2003.

[76] Verstichel, J. and Berghe, G. V. A late acceptance algorithm for the lock scheduling problem. *Logistik Management*, pages 457–478, 2009.

[77] Willemen, R. *School Timetable Construction: Algorithms and Complexity*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[78] Wright, M. School timetabling using heuristic search. *Journal of Operational Research Society*, 47:347–357, 1996.

[79] Yang, Z., Chu, F., and Chen, H. A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research*, 221(3):521 – 532, 2012.