Timetabling at High Schools

Sørensen, Matias; Stidsen, Thomas Jacob Riis; Herold, Michael B. ; Pisinger, David

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Ph.D. thesis

Timetabling at High Schools

Matias Sørensen

Management Science
Department of Management Engineering
Technical University of Denmark
Produktionstorvet, Building 426
DK-2800 Kgs. Lyngby
Denmark

MaCom A/S
Vesterbrogade 48, 1.
DK-1620 Kbh. V
Denmark

November, 2013

Danish title:
Skemalægning på gymnasiale uddannelsesinstitutioner
Type: Ph.D. thesis

Author: Matias Sørensen, msso@dtu.dk, ms@macom.dk

ISBN-nr: ??

Management Science
Department of Management Engineering
Technical University of Denmark
Produktionstorvet, Building 426
DK-2800 Kgs. Lyngby
Denmark
Phone: +45 45 25 48 00, Fax: +45 45 25 48 05
phd@man.dtu.dk

MaCom A/S
Vesterbrogade 48, 1.
DK-1620 Kbh V.
Denmark
Phone: +45 33 79 79 00

November, 2013

Abstract

High school institutions face a number of important planning problems during each school-year. This Ph.D. thesis considers two of these planning problems: The High School Timetabling Problem (HSTP) and the Consultation Timetabling Problem (CTP). Furthermore a framework for handling various planning problems is considered, known as the Generalized Meeting Planning Problem (GMPP). The view taken on these problems is that they are mathematical optimization problems, where the goal is to find the optimal solution (from the set of all feasible solutions). This view allows state-of-the-art methods from the field of Operations Research to be applied.

This thesis is composed of three parts. The first part introduces the relevant methodologies of Operations Research, describes the considered optimization problems, summarizes the scientific articles and lists the scientific contributions of the thesis. The second part contains the main scientific papers composed during the Ph.D. study. The third part of the thesis also contains scientific papers, but these are included as an appendix.

In the HSTP, the goal is to obtain a timetable for the forthcoming school-year. A timetable consists of lectures scheduled to time-slots, and each lecture has a number of resource requirements. The goal is to obtain a schedule such that the individual timetable for each resource fulfills a number of requirements. Two versions of the HSTP are considered: The Generalized High School Timetabling Problem (GHSTP) (based on the publicly available XHSTT format for modeling instances and solutions of the HSTP) and the Danish High School Timetabling Problem (DHSTP). For both problems a complex Mixed-Integer Programming (MIP) model is developed, and in both cases are empirical tests performed on a large number of real-life datasets, which show that the MIP model is a challenge to solve for a state-of-the-art generic MIP-solver. A heuristic based on Adaptive Large Neighborhood Search (ALNS) is developed for the GHSTP, and this heuristic was part of the final round of International Timetabling Competition 2011 (ITC2011). An ALNS heuristic is also developed for the DHSTP, and computational results show that this is currently the best known solution algorithm. Furthermore, the thesis shows the relation between the GHSTP and the DHSTP, and instances of the DHSTP are made publicly available in the XHSTT format.

An extension of the Two-Stage Decomposition (TSD) method is also shown in this thesis, which makes the TSD capable of handling both the GHSTP and the DHSTP. In a TSD approach, a MIP model is split into two separate models which are solved in sequence, while maintaining optimality (as far as possible). This reduces the total amount of variables significantly compared to the original MIP model. Whether or not the TSD is an exact solution method in the context of GHSTP and DHSTP is determined by certain characteristics of a given dataset. For both the GHSTP and the DHSTP, the TSD is capable of producing lower bounds, even though it might not be an exact method for the dataset at hand. For the DHSTP, the TSD is shown to be theoretically capable of producing near-optimal solutions for an arbitrary dataset, and computational results show that the TSD provides both better solutions and better bounds than the original MIP model. For the GHSTP, the TSD is an exact method for the majority of the

considered datasets. However, in this case the computational results do not clearly show that the TSD outperforms the original MIP model.

An algorithm hybridizing MIP and metaheuristics is developed and applied to both the GHSTP and the DHSTP. This algorithm is part of the recent trend called matheuristics, which is a promising class of solution approaches for many types of optimization problems. In the implemented matheuristic, a MIP solver is used as a low-level search mechanism, and an adaptive layer of the algorithm guides the search on the overall level. In terms of the GHSTP, this matheuristic has shown to be competitive with the winner of Round 2 of ITC2011. For the DHSTP, the algorithm outperforms the exact approaches, but not the ALNS algorithm when compared on a low time-limit. Given a time-limit of two hours, the matheuristic obtains solutions which are within 15.2% of optimum in average for 100 real-life dataset of the DHSTP.

The CTP has not been described in the scientific literature before. The problem consists of scheduling meetings between a single student and a number of teachers to time-slots. The primary aim of a meeting is to allow the teachers to provide feedback to the student w.r.t. educational progress. A proof of $\mathcal{NP}$-hardness is given, and a MIP model is developed. Also for this problem, an ALNS heuristic is shown to perform very well, producing solutions that are within 3% of optimum in average on 200 real-life datasets.

The GMPP is a framework for handling a number of different planning problems. The goal of the GMPP is to schedule meetings between certain resources to time-slots, such that no resource attends more than one meeting at any time. A model of the problem is proposed which is based on a Column Generation approach. A key feature of this model is that most problem-specific constraints are handled by the subproblem of the Column Generation algorithm. This leads to a Branch-and-Price algorithm for the GMPP which is independent of these problem-specific details, but still applicable to a range of optimization problems. As a test-case for the GMPP, the CTP is used. The Branch-and-Price algorithm obtains solutions within 3% of optimum in average on the same set of instances as the ALNS algorithm.

The thesis show that real-world high school timetabling problems are a challenge to solve for exact methods, even with the recent advances of generic MIP solvers and when applying state-of-the-art techniques such as TSD. In a practical setting for these problems, the tests performed in this thesis show that heuristics in general produce the best solutions. However, exact methods which can provide bounds on optimum are valuable for evaluating the performance of the heuristics. For the CTP, the performance of an exact algorithm (the Branch-and-Price algorithm) is competitive with the performance of the tested heuristic (the ALNS heuristic). This shows that it is possible to use exact methods for CTP in a practical setting.

Resumé

In Danish

Gymnasiale uddannelsesinstitutioner skal løse en række vigtige planlægningsproblemer i løbet af et skoleår. Denne ph.d. afhandling omhandler to af disse planlægningsproblemer: Skemalægnings-problemet (SP) og Konsultations-planlægningsproblemet (KPP). Desuden betragtes det Generaliserede møde-planlægningsproblem (GMPP), som kan håndtere en lang række forskellige planlægningsproblemer. Planlægningsproblemerne anskues i denne afhandling som matematiske optimeringsproblemer, hvor målet er at finde den optimale løsning (ud fra sættet af alle mulige løsninger). Ved at anskue problemerne på denne måde er det muligt at anvende førende løsningsmetoder indenfor den videnskabelige disciplin operationsanalyse.

Afhandlingen består af tre dele. Den første del introducerer de relevante metodologier inden for operationsanalyse, beskriver de betragtede optimeringsproblemer, opsummerer de videnskabelige artikler, samt gennemgår afhandlingens videnskabelige bidrag. Den anden del af afhandlingen indeholder de primære videnskabelige artikler som er udarbejdet i løbet af ph.d. studiet. Den tredje del af afhandlingen indeholder også videnskabelige artikler, men disse er inkluderet som et appendiks.

SP omhandler planlægningen af det årlige skema, hvor alle lektioner skal tildeles en skema-position og et sæt af ressourcer således, at det individuelle skema for hver ressource opfylder et antal kriterier. To forskellige versioner af SP betragtes: Det generelle skemalægnings-problem (GSP) (baseret på det offentligt tilgængelige XHSTT format til at modellere instanser og løsninger af SP) og det danske skemalægnings-problem (DSP). For begge problemer vises en heltal-programmeringsmodel, og i begge tilfælde udføres empiriske tests på et stort antal realistiske datasæt, hvilket viser at modellerne er en udfordring for en førende generisk heltal-programmerings løsningsalgoritme. For GSP udvikles der en heuristik baseret på *Adaptive Large Neighborhood Search* (ALNS). Denne heuristik var en del af finalerunden ved *International Timetabling Competition 2011* (ITC2011). Endvidere udvikles også en ALNS heuristik til DSP, som p.t. er den bedst kendte løsningsalgoritme. Ydermere vises relationen mellem GSP og DSP, og instanser af DSP gøres offentligt tilgængelige i XHSTT formatet.

En udvidelse af metoden To-Fase Dekomponering (TFD) vises, hvilket gør TFD i stand til at håndtere både GSP og DSP. I en TFD metode deles heltal-programmerings modellen i to separate modeller, og disse to modeller løses sekventielt, således at optimalitet bevares (så vidt som muligt). Denne fremgangsmåde reducerer betydeligt det totale antal af variable i modellen. Hvorvidt TFD er en eksakt løsningsmetode afhænger af forskellige karakteristika ved det enkelte datasæt. For både GSP og DSP er TFD i stand til at finde nedre grænser for optimum, også selvom metoden i sig selv ikke er eksakt mht. det betragtede datasæt. I tilfælde af DSP vises det, at TFD er teoretisk nær-optimal for et arbitrært datasæt og empiriske resultater viser, at TFD er i stand til at finde både bedre løsninger og bedre nedre grænser for optimum end den originale heltal-programmerings model. For GSP vises det, at TFD er en eksakt metode for

hovedparten af de betragtede datasæt. Dog viser de empiriske resultater ikke klart at TFD er en bedre løsningsmetode end den originale heltal-programmerings model.

En hybrid-algoritme baseret på heltal-programmering og metaheuristikker er vist, og anvendt på både GSP og DSP. Denne algoritme er en del af den nye trend *matheuristics*, som er en lovende type løsningsalgoritme til mange optimeringsproblemer. En heltal-programmerings løsningsalgoritme bruges som den basale søgemekanisme, og et adaptivt lag guider søgningen på det overordnede niveau. Til GSP viser denne algoritme sig at være konkurrencedygtig med vinderen af runde 2 i finalerunden ved ITC2011. Til DSP viser algoritmen sig at være bedre end de eksakte løsningsmetoder, men ikke bedre end ALNS algoritmen når givet en kort tidsbegrænsning. Ved en tidsbegrænsning på to timer, så er matheuristic algoritmen i stand til at finde løsninger som i gennemsnit er indenfor 15,2% af optimum når 100 realistiske problem-instanser af DSP betragtes.

KPP er ikke blevet beskrevet i den videnskabelige litteratur før. Problemet består i at tildele tidspunkter til møder mellem en elev og et antal lærere. Det primære mål med det enkelte møde er, at lærerne får mulighed for at give eleven feedback mht. vedkommendes faglige udvikling. Et bevis for at problemet er $\mathcal{NP}$-hårdt vises, og der udvikles en heltal-programmeringsmodel. Også for dette planlægningsproblem opnår en ALNS heuristik gode resultater, da den er i stand til at producere løsninger som i gennemsnit er indenfor 3,0% af optimum når 200 realistiske problem-instanser betragtes.

GMPP er en problem-ramme til at håndtere en række forskellige planlægningsproblemer. Formålet med GMPP er at tildele møder mellem en række forskellige ressourcer til tidspunkter således, at ingen ressourcer har mere end ét møde i hvert tidspunkt. Der foreslås en model af problemet, som er baseret på en *Column Generation* løsningsmetode. En vigtig egenskab ved modellen er, at de fleste problem-specifikke begrænsninger bliver håndteret i under-problemet i *Column Generation* algoritmen. Dette leder til en *Branch-and-Price* algoritme til KPP som er uafhængig af disse problem-specifikke detaljer, men som kan anvendes på en lang række planlægningsproblemer. Som en test af GMPP bruges KPP. I dette tilfælde er Branch-and-Price algoritmen i stand til at finde løsninger, som er indenfor 3,0% af optimum i gennemsnit over de samme problem-instanser som ALNS algoritmen blev anvendt på.

Denne afhandling viser at realistiske skemalægningsproblemer er svære at løse for eksakte metoder, selv med de seneste fremskridt indenfor generiske heltal-programmerings løsningsalgoritmer, og når state-of-the-art tekniker anvendes (f.eks. TFD). I praksis er heuristiske løsningsalgoritmer i stand til at producere bedre løsninger. Eksakte metoder der kan give grænser for optimum er dog værdifulde når kvaliteten af de heuristiske løsninger skal evalueres. I tilfælde af CTP er en eksakt metode (*Branch-and-Price* algoritmen) konkurrencedygtig med den testede heuristik (ALNS heuristikken). Dette viser at det er muligt at bruge eksakte metoder til KPP i praksis.

Preface

The studies presented in this thesis document in part the fulfillment of the requirements for the degree of Ph.D. at the Department of Management Engineering, Technical University of Denmark. The Ph.D. project ran from December 2010 to November 2013. The project was supervised by Thomas R. Stidsen (main supervisor), Michael B. Herold (company supervisor) and David Pisinger (co-supervisor).

This Ph.D. project is part of a research project which considers timetabling problems at educational institutions. A considerable amount of the studies have been performed jointly with Ph.D. student Simon Kristiansen from The Department of Management Engineering, Technical University of Denmark.

This Ph.D. study was performed under the *Industrial Ph.D. Programme*, in which the Ph.D. candidate is both employed in a private company and enrolled at a university. An Industrial Ph.D. project should contribute to the perspectives of the company, as well as fulfilling the requirements of the Ph.D. school. In this Ph.D. project the commercial product Lectio has been used as an interface for implementation. Lectio is available to high school institutions in Denmark, and is used by the majority of these to handle a large variety of administrative and operational tasks.

Matias Sørensen

Kgs. Lyngby, Denmark, November 2013

Acknowledgments

I would like to thank a number of people who helped me in various ways during this Ph.D. project. First of all, thank you to both the division of Management Science, The Department of Management Engineering, Technical University of Denmark, and MaCom A/S for giving me the opportunity to pursue a Ph.D. degree. I would like to thank all my colleagues who supported my studies for the past three years, especially Simon Kristiansen for our intensive and productive cooperation.

Thank you to all my supervisors for the good discussions we had throughout the project, and to Niels-Christian F. Bagger and Jørgen T. Haahr for proof reading some chapters in this thesis.

During the project I visited The Chair of Operations Research, RWTH Aachen University. Thank you to everybody at the department for your hospitality during my stay, especially Florian Dahms and Prof. Dr. Marco Lübbecke.

Finally I would like to thank Marie for your overwhelming support throughout the entire project.

Contents

Part I

Introduction

Abbreviations

ALNS	Adaptive Large Neighborhood Search
B&P	Branch-and-Price
CG	Column Generation
CTP	Consultation Timetabling Problem
DHSTP	Danish High School Timetabling Problem
GHSTP	Generalized High School Timetabling Problem
GMPP	Generalized Meeting Planning Problem
HSTP	High School Timetabling Problem
IP	Integer Programming
ITC2011	International Timetabling Competition 2011
LNS	Large Neighborhood Search
LP	Linear Program
MIP	Mixed-Integer Programming
OR	Operations Research
PCTP	Parental Consultation Timetabling Problem
SCTP	Supervisor Consultation Timetabling Problem
TSD	Two-Stage Decomposition

Chapter 1

Thesis Background

The research area *Educational Timetabling* is defined as the timetabling problems originating from educational institutions such as high schools and universities. Kingston (2013a) divides the area into several sub-fields, where the most prominent are high school timetabling, examination timetabling and university course timetabling. This thesis mainly considers high school timetabling, denoted the *High School Timetabling Problem* (HSTP) in the following, but also another important planning problem is studied, the *Consultation Timetabling Problem* (CTP). These planning problems originate from high school institutions and they are a challenge to solve for the high school administration. Furthermore a framework for handling various timetabling problems is considered, known as the Generalized Meeting Planning Problem (GMPP).

The view taken on the timetabling problems is that they are *optimization problems*, where the goal is to find the optimal solution (from the set of all feasible solutions). This view allows for applying structured solution methods, which takes advantage of the power of modern computers. Optimization problems are a subfield of *Operations Research* (OR), and OR methods are used throughout this thesis. OR can informally be defined as *a discipline that deals with the application of advanced analytical methods to help make better decisions* (INFORMS, 2013), and is usually seen as the intersection of the disciplines computers science and mathematics. For additional information about OR, see Section 1.1.

The application of OR techniques to optimization problems has several advantages: 1) The ability to find the optimal solution for an optimization problem can lead to improved efficiency, where the definition of efficiency is problem-dependent. 2) Assuming that the given solution algorithm is faster than its alternatives; its application can reduce the amount of resources required to solve the problem. 3) In case the solution algorithm is capable of providing several different solutions, the owner of the optimization problem can choose the most preferable one of these. The timetabling problems considered in this thesis are realistic planning problems which contain all requirements needed in practice by the high schools, and the instances used to establish empirical computational results are non-simplified. Some of the presented solution methods have been deployed to a large number of end-users at the high schools, which show the practical usability of the algorithms. Thereby the high schools have access to improved tools for solving these timetabling problems.

Some of the considered optimization problems are specific to high schools in Denmark. In this context the term 'high school' is used rather broad, and covers institutions offering the following *upper secondary education programs*: STX, HHX, HTX, and HF (The Ministry of Education in Denmark, 2013). STX, HHX and HTX take three years to complete for a student, and HF takes two years to complete. In Denmark there are 146 institutions offering STX and/or HF, 60

offering HHX, and 38 offering HTX. The Ministry of Education in Denmark (2009) claims that the savings-potential of a more throughout digitalizing of the timetabling process for institutions offering upper secondary educations in Denmark is 100,000,000 DKK (≈ €13,400,000). This supports the claim that efficient solution algorithms are important tools for the high schools.

1.1 Operations Research

This section gives a short introduction to OR, and briefly describes the methodologies relevant for this thesis.

In OR, one typically uses advanced analytic mathematical method to analyze a given decision-problem, and thereby determine the optimal solution to this problem (Taha, 1997, p. 2). The range of applications of OR is very wide, and includes decision-problems such as supply chain management, crew schedule planning in the airline industry, transport logistics for trucks, cost reduction in production scheduling, design of telecommunications networks, financial engineering, and facility location problems considering the placement of factories. Decision-problems are usually described in terms of a mathematical model, which involves variables and relationships between these variables (Rardin, 1998, p. 4). The variables usually model specific decisions, and the model may contain an objective function which is capable of measuring the overall impact of the current set of decisions (the values of the variables). Thereby the optimal solution to the mathematical model will also define an optimal set of decisions for the decision-problem.

It is common for an optimization problem to contain both *hard constraints* and *soft constraints*. Hard constraints model conditions for the feasibility of a solution, such that solutions which obey all hard constraints are said to be feasible. Soft constraints model different preferences for the characteristics of a solution, and the objective value of solution is usually determined in terms of the (weighted) violation of the soft constraints. The concept of soft- and hard constraints is used intensively throughout this thesis.

1.1.1 Integer Programming

A methodology commonly used within OR is *Integer Programming* (IP). This is a subfield of mathematical optimization where the variables are required to take integer values. An IP model of an optimization problem contains a set of constraints which restrict the possible values the variables can take, and an objective function which is sought to be minimized (or maximized). Throughout this thesis, all considered IP models are implicitly understood to be *integer linear programs*, meaning that all terms of the problem are linear, except for the integer requirements. In case only a subset of the variables are required to take integer variables (and thereby the remaining variables are defined over continuous range), the term *Mixed-Integer Programming* (MIP) is used. Throughout this thesis the term MIP is used even though a given model might not contain any continuous variables (i.e. all variables are integer variables). Mathematical optimization methods can be applied to a MIP model, meaning that the global best solution is sought. Methods capable of finding optimal solutions are also known as *exact methods*.

A bound on the objective value of an optimal solution of a MIP model can be obtained by relaxing all integer requirements on the variables (such that a pure linear system of inequalities is obtained), and thereby solve the resulting *Linear Program* (LP). This is known as the LP-relaxation. Solving the LP-relaxation of a MIP model is usually much easier than solving the MIP model itself. Throughout this thesis *generic MIP solvers* are used intensively, typically in a black-box fashion. By generic is meant that these solvers take any MIP model as input, and aim at performing well across a wide range of models. A key point in this context is that MIP

solvers use the LP-relaxation internally, and attempts to close the gap between the objective of the LP-relaxation and the objective of the found integer solution (in case the gap is closed the optimal solution is found). So even though the MIP solver might not find the optimal solution within the given resource limitations, it can provide a bound on optimum as well as a sub-optimal solution. The progress for generic MIP solvers has been considerable in the last decades. Bixby (2012) argues that the runtime of the generic solver CPLEX has been improved with a factor 29000 since its initial release in 1991 and until version 11.0 released in 2007 (not including the technical advances of computers).

This thesis also treats decomposition methods of MIP models, which can provide more efficient solution methods by dividing the optimization problem into different parts. This includes *Branch-and-Price* (B&P) algorithms, which is based on the *Column Generation* (CG) method for solving LPs. A description of these methods is given in Chapter 3. Another type of decomposition is *Two-Stage Decomposition* (TSD), which is a recent innovative method of Lach and Lübbecke (2008, 2012). In its present form, its application is closely tied with the structure of timetabling problems; however this might change in the future. TSD is a promising branch of exact methods for timetabling, and an in-depth description of its application to HSTP is given in Section 2.3.

1.1.2 Heuristics

Heuristics are another type of solution method for optimization problems, which are based on applying rules of thumb to produce a good solution to a given optimization problem (Taha, 1997). Heuristics are typically less sensitive to the size of the input-data compared to solution approaches based on MIP, and are therefore usually used in a practical setting when computational resources are scarce (such as a low timelimit). However, heuristics generally provide no bounding information, and can therefore not issue any certificates of optimality like MIP methods can. Heuristics therefore reside on the fact that they perform well for the given optimization problem, but rely on external information to evaluate their performance. Such external information usually comes in the form of bounds provided by exact methods.

Metaheuristics is a sub-field of heuristics, and are defined as "...an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space." (Osman and Laporte, 1996). Possibly the most popular type of subordinate heuristic is *local search*. A local search heuristic starts from an initial solution and examines solutions in the neighborhood (usually defined as solutions which can be found by simple perturbations of the current solution). If one of these neighbor solutions improves the current solution, the neighbor solution is *accepted* and replaces the current solution, and the procedure is repeated. The algorithm terminates when no improving solution can be found, thereby ending in a local optimum (Osman and Laporte, 1996). An improved algorithm embeds this local search scheme in a metaheuristic, for instance a *Simulated Annealing* framework which allows the acceptance of worse solutions with a certain probability (Laarhoven and Aarts, 1987). This makes the algorithm capable of escaping local optimum, which can lead to better solutions.

1.1.2.1 Large Neighborhood Search

The most used metaheuristic in this thesis is based on *Large Neighborhood Search* (LNS), which iteratively *destroys* and *repairs* (parts of) a solution, using problem-specific methods. The goal is to incrementally improve the solution, by letting the destroy operation target parts of the solution which can possibly be improved by the repair operation. The variant of LNS mainly considered throughout this thesis is *Adaptive Large Neighborhood Search* (ALNS), usually credited to Pisinger and Ropke (2005); Ropke and Pisinger (2006). In this variant of the heuristic,

multiple destroy and repair methods are used, and their performance is tracked throughout the solution approach. In every iteration, a selection of a destroy method and a repair method is performed on the basis of certain performance-indicators obtained in the previous iterations of the procedure. The goal of this selection is to favor the use of destroy and repair methods which have previously performed 'good'. ALNS heuristics have been used with success for various types of optimization problems, most prominent variants of the vehicle routing problem (see e.g. Azi et al. (2010); Laporte et al. (2010); Salazar-Aguilar et al. (2011); Ribeiro and Laporte (2012)), but also lot-sizing (Muller et al. (2011)), machine scheduling (Wang et al. (2012)), and others.

Due to the extensive use of ALNS heuristics throughout this thesis, its relationship with other metaheuristics is briefly surveyed in the following. Ahuja et al. (2002) describe the class of *Very Large-Scale Neighborhood Search* algorithms, and Pisinger and Ropke (2010) classify LNS algorithms as belonging to this class. Very Large-Scale Neighborhood Search algorithms are defined as having the property that the searched neighborhood grows exponentially with the instance size or the property that the neighborhood is too large to be searched explicitly in practice (Pisinger and Ropke, 2010). Other types of Very Large-Scale Neighborhood Search algorithms are mentioned in the following: *Variable Depth Neighborhood Search* considers neighborhoods of increasing size, using only large neighborhoods in case none of the smaller neighborhoods can improve the current solution. *Variable Neighborhood Search* usually operates on structurally different neighborhoods (Hansen and Mladenović, 2001).

Another branch of heuristics which is related to ALNS is *Hyper-heuristics*, which tries to raise the level of generality at which search methodologies can operate (Burke et al., 2003). Recently, Burke et al. (2010) defined hyper-heuristics as follows: *A hyper-heuristic is a search method or learning mechanism for selecting or generating heuristics to solve computational problems.* Furthermore, they describe two classes of hyper-heuristics: (I) *Heuristic selection methodologies* which combine pre-existing low-level heuristics, and (II) *Heuristic generation methodologies* which generate new heuristic methods using basic-components (which can for instance be low-level heuristics). Furthermore, w.r.t. *learning* by feedback from the search process, Burke et al. (2010) distinguishes between *offline learning* (learn by knowledge obtained on a set of training instances) and *online learning* (learning takes place while solving an instance of a problem). Given these classifications, the relationship with the ALNS methodology is discussed in the following: Applying a destroy method and a repair method in each iteration (and thereby combining these methods) can be seen as the generation of a new heuristic, and hence falls into classification (II). In terms of feedback, this is a central process in an ALNS algorithm. The feedback received throughout the algorithm is used to increase the probability of selecting well-performing destroy/repair methods, which is an online learning process. Notice that offline learning can also be used in the tuning process for certain parameters of the algorithm.

1.1.3 Matheuristics

This thesis also uses algorithms based on hybridization of exact methods and heuristics, usually denoted *matheuristics* in a broader sense (by the contraction of *mathematical optimization* and *heuristics*). Informally, it is the goal of this class of methods to combine the properties of exact methods with the rapidness of heuristics, and thereby *enjoy the best from both worlds* (Ryan, 2012). The combining of these two classes of methodologies is still in its infancy, but has a large potential for future applications according to (Maniezzo et al., 2009a,b). Blum et al. (2011) describe the motivation behind hybridizing of algorithms as follows: *...to exploit the complementary character of different optimization strategies, that is, hybrids are believed to benefit from synergy.*

Caserta and Voss (2010) describe the hybridization of exact methods and metaheuristics in terms of a master-slave structure. That is, either (I) the metaheuristic acts as at a higher level and controls the exact approach or (II) the exact method acts as the master and controls the metaheuristic. In algorithms of type (I), the searched neighborhood is typically composed of only a small part of the solution space, but is searched by the exact approach. The job of the metaheuristic is then to guide the search on an overall level. Notice that this type of algorithm resembles the concepts of an ALNS algorithm (apart from the use of exact methods), since an exponentially large neighborhood is searched and the approach is guided heuristically (the adaptive layer in the ALNS algorithm). Examples of algorithms of type (II) could be MIP-solvers, which are dependent on internal heuristics for speeding up the solution process. Having incumbent solutions of good quality early on in the process can help in pruning nodes in the branch-and-bound tree, and thereby reduce the computational effort. In this respect, Lodi (2013) argues that modern MIP solvers are heuristic in nature.

1.2 Thesis Reading Guide

The literature considering educational timetabling problems by OR methods is mainly concerned with heuristics. A contribution of this thesis is the application of exact methods to educational timetabling problems. These exact methods are all based on IP, and generally involve large MIP models with many diverse terms in the objective function and a large variety of constraints. This complexity arises from the fact that the considered timetabling instances are non-simplified real-world optimization problems which are used in practice by a large amount of high schools. Therefore the OR model of the timetabling problem can be thought of as an abstract model which encapsulates all requirements of the high schools. This inevitably gives rise to complicated models, which in turn yields complex MIP models.

Another contribution of this thesis is the development of several solution methods for the considered optimization problems. An important concept in OR is the separation of the optimization problem from the solution method, so these solution methods can typically be read without an in-dept review of the complex models (although some basic knowledge is typically required).

The thesis is divided into three parts. Part I considers the introductory topics. Chapter 2 of Part I describes the main optimization problem of this thesis, namely the HSTP. Section 2.1 considers the GHSTP and Section 2.2 considers the DHSTP. Chapter 3 describes the GMPP and the CTP. In Chapters 2 and 3 the relationship between the optimization problems is also discussed, including comparisons of the MIP models. Chapter 4 gives an overview of the results obtained throughout the Ph.D. study, including a brief summary of the scientific papers and the scientific contributions. Finally, Chapter 5 concludes on the findings and discusses subjects for future research.

Part II of the thesis contains the main scientific papers produced throughout the Ph.D. study, which is the scientific background for the material in Part I. Part III contains additional papers as an appendix.

Chapter 2

High School Timetabling

This chapter describes the *High School Timetabling Problem* (HSTP), which is the main topic of the thesis. Two versions of the HSTP are considered, the *Generalized High School Timetabling Problem* (GHSTP), and the *Danish High School Timetabling Problem* (DHSTP). These are described in Section 2.1 and in Section 2.2, respectively. The goal of these sections is to describe the problems in detail and show how the problems relate to each other. A basic description covering the fundamental elements of both problem-versions is given in the following paragraphs.

The HSTP concerns the constructing of a timetable for the forthcoming school-year at a specific high school. This is a problem faced by high schools around the world at least once per year. The specific characteristics of the HSTP vary greatly depending on the country and the school from which it originates, due to both differences in the educational structures among countries and different ways of organizing the teaching at each school. In general the problem is hard to solve, which makes the solution process a time consuming task for the high school administration, and obtaining solutions of high quality might be out of scope. Applying OR techniques can potentially reduce the time required to solve the problem and produce solutions of higher quality than obtained by other methods (for instance manual methods).

Formally, the HSTP can be described as follows. The set of events $\mathcal{E}$ models the lectures which are to be scheduled. The set of resources is denoted $\mathcal{R}$, and each resource is of a specific type, e.g. room, student or teacher. Each event $e \in \mathcal{E}$ requires a subset of resources $A_e \subseteq \mathcal{R}$ (*fixed resources*), or requires that a number of resources of a certain type are assigned to it (*free resources*). An event is required to be assigned a time-slot. The set of time-slots is denoted $\mathcal{T}$, and is build by dividing each day into a number of discrete non-overlapping time-slots. Throughout the rest of the thesis the terms *time-slot* and *time* are used interchangeably. An important constraint is to perform the assignment of events to times such that no clashes among resources occur, meaning that no resource is assigned more than one event in each time-slot. Various other hard- and soft-constraints are typically imposed as well, and most of these define requirements and preferences for the resources. For instance, a teacher might have different priorities for the times he or she is assigned to teach.

Even though the HSTP is concerned with a yearly timetable, the planning is usually performed w.r.t. one or two weeks only. This is based on the assumption that no differences exist among weeks throughout the school-year, i.e. holidays and other irregular activities are assumed to not exist. Once a satisfactory timetable has been found for this simpler problem, this timetable is used as a basic solution for every week of the year. The weekly irregularities can then be incorporated by performing a (hopefully small) number of perturbations for each week. The advantage of this approach is that the yearly timetabling problem can be solved by solving a

series of smaller problems. This thesis deals with the problem of constructing the basic timetable for the short time-horizon (which is denoted the HSTP).

2.1 The Generalized High School Timetabling Problem

The GHSTP is based on the XHSTT format (see Post et al. (2012a) for an in-depth description of the format), which seems to be the only active format for exchanging problem-instances and corresponding solutions for high school timetabling. The format is based on the *eXtensible Markup Language* standard, and a large number of instances is publicly available at the XHSTT website (Post, 2013b) (currently there are around 50 instances from 11 different countries available). Having a standard format for modeling problem instances is an advantage for the high school timetabling community, as it facilitates the exchange of knowledge, for instance in terms of a common ground for comparing solution algorithms. However, the format was developed recently, and some effort is still required to make it more widely known within the community. The thoroughness of the XHSTT format results in a model of the GHSTP which is complex. In this section the most fundamental properties of the GHSTP is described and a MIP model is shown which contains the basic constraints. The full model can be found in Paper A.

Based on the basic description of the HSTP, a description of the GHSTP is given in the following. The GHSTP contains a number of additional properties and constraints. Each event $e \in \mathcal{E}$ has a duration $D_e \in \mathbb{N}$, which denotes the number of times which the event spans. An event defines a number of event resources, which each is a requirement for a certain resource (*fixed resources*), or a request to be assigned a resource of a certain type (*free resources*). An event resource of event $e \in \mathcal{E}$ is indexed by $er \in e$. An applicable resource for event resource er is indexed by $r \in er$.

An important concept of XHSTT is the splitting of events into smaller pieces, as described in the documentation (Kingston, 2013c): *A solver (*for XHSTT*) is expected to split some of the events into smaller pieces. This allows courses to spread their lessons through the cycle, without forcing the durations of those lessons to be fixed in advance, as would be the case if each lesson was modeled as a distinct event.* To model the splitting of events, the concept of *sub-events* is introduced. The set of sub-events is denoted $\mathcal{SE}$, and a sub-event of an event $e \in \mathcal{E}$ is denoted $se \in e$. The duration of a sub-event is denoted $D_{se} \in \mathbb{N}$. A sub-event inherits all resource requirements defined by the event. The set $\mathcal{SE}$ it build by enumerating all possible sub-events for all events, which amounts to a total of $\sum_{e \in \mathcal{E}} \sum_{i=1}^{D_e} \lfloor \frac{D_e}{i} \rfloor$ elements. For instance, there exist a total of five sub-events for an event with a duration of 3, and these have durations 1,1,1,2 and 3, respectively. By XHSTT definition, the total sum of the *active* sub-events of an event should equal the total duration of the event. A sub-event is active iff it is assigned a time or a resource which is not fixed. To allow a sub-event to be inactive (which is required for feasibility of the MIP model), the sets of times and resources are extended by a dummy-element, denoted t_D and r_D, respectively.

The XHSTT format includes several different types of constraints (see Table 2.1), which can be used to model various requirement on the instance level (i.e. an instance of XHSTT is not required to use all types of constraints). This allows instances of very diverse character of the HSTP to be modeled in a uniform way. Denote by $\mathcal{C}_G$ the set of all constraints native to the XHSTT format. With each constraint $c \in \mathcal{C}_G$ is associated a weight $w_c \in \mathbb{N}^+$, a specific way to derive the violations incurred, and a method to convert these violations into a cost, denoted a *CostFunction*. A constraint $c \in \mathcal{C}_G$ defines a set of point-of-applications, for instance a set of events or a set of resources. A point-of-application defines an entry for which an amount of violation can be calculated. In the following a point-of-application for constraint $c \in \mathcal{C}_G$ is indexed

by $p \in c$. For constraint $c \in \mathcal{C}_G$ and point-of-application $p \in c$ a variable $s_{c,p} \in \mathbb{N}_0$ is defined, which denotes the amount of violation. This variable is given as input to the CostFunction, which converts the violations into a cost $\in \mathbb{N}_0$. Five different CostFunctions exist, the most simple being `Sum`, which simply sums the violation of every point-of-application. The other types of CostFunction are nonlinear measures, which can be calculated using auxiliary variables (see Paper A provides a description). The contribution of a constraint $c \in \mathcal{C}_G$ to the objective is $w_c \cdot \text{CostFunction}(s_{c,p})$. Furthermore, a constraint can either be defined as a hard or a soft constraint. Therefore the objective value of a XHSTT solution consists of two values, the cost of the violation of the hard constraint (the *hard cost*) and the cost of the violation of the soft constraints (the *soft cost*), and is usually written as (hard cost, soft cost). Allowing the violation of hard constraints is incompatible with the usual definition of these, but this is how the XHSTT format is defined.

Table 2.1: Constraint types of the XHSTT format (Post et al., 2012b). See Kingston (2013b) for more details.

Constraint	Description
Assign Resource	Event resource should be assigned a resource
Assign Time	Event should be assigned a time
Split Events	Event should split into a constrained number of sub-events
Distribute Split Events	Event should split into sub-events of constrained durations
Prefer Resources	Event resource assignment should come from resource group
Prefer Times	Event time assignment should come from time group
Avoid Split Assignments	Set of event resources should be assigned the same resource
Spread Events	Set of events should be spread evenly through the cycle
Link Events	Set of events should be assigned the same time
Order Events	Set of events should be ordered
Avoid Clashes	Resource's timetable should not have clashes
Avoid Unavailable Times	Resource should not be busy at unavailable times
Limit Idle Times	Resource's timetable should not have idle times
Cluster Busy Times	Resource should be busy on a limited number of days
Limit Busy Times	Resource should be busy a limited number of times each day
Limit Workload	Resource's total workload should be limited

To formulate a MIP model of the GHSTP, a number of decision variables are required. Let variable $x_{se,t,er,r} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is placed in time (the starting time) $t \in \mathcal{T}$ and resource $r \in er$ is assigned to event-resource $er \in se$, and 0 otherwise. Notice that this variable decides the starting time of the sub-event, and the duration of the sub-event determines the amount of contiguous times which the sub-event spans. Let variable $y_{se,t} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is assigned to starting time $t \in \mathcal{T}$, and 0 otherwise. Let variable $w_{se,er,r} \in \{0,1\}$ take value 1 if event resource $er \in se$ of sub-event $se \in \mathcal{SE}$ is assigned resource $r \in er$, and 0 otherwise. Variable $u_{se} \in \{0,1\}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is active, and 0 otherwise.

The MIP model of the GHSTP is shown in Model (2.1). The native constraints $\mathcal{C}_G$ of the XHSTT format are not modeled here, as they are not necessary for relating the GHSTP to the other timetabling problems of this thesis. A formulation of all native XHSTT constraints in context of Model (2.1) can be found in Paper A. Therefore it can be said that Model (2.1) implicitly contains the native XHSTT constraints (as indicated by Constraint (2.1i)).

GHSTP MIP model (2.1)

$$\min \quad \sum_{c \in \mathcal{C}_G} w_c \cdot \text{CostFunction}(s_{c,p}) \tag{2.1a}$$

s.t.

$$\sum_{t \in \mathcal{T}, r \in er} x_{se,t,er,r} = 1 \qquad \forall se \in \mathcal{SE}, er \in se \tag{2.1b}$$

$$\sum_{er \in se, r \in er} x_{se,t,er,r} = |er|_{se} \cdot y_{se,t} \qquad \forall se \in \mathcal{SE}, t \in \mathcal{T} \tag{2.1c}$$

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \qquad \forall se \in \mathcal{SE}, er \in se, r \in er \tag{2.1d}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} \leq u_{se} \qquad \forall se \in \mathcal{SE} \tag{2.1e}$$

$$\sum_{r \in er \setminus r_D} w_{se,er,r} \leq u_{se} \qquad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{2.1f}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} + \sum_{\substack{r \in er \setminus r_D \\ er \in se, PA_{er} = 0}} w_{se,er,r} \geq u_{se} \qquad \forall se \in \mathcal{SE} \tag{2.1g}$$

$$\sum_{se \in e} D_{se} \cdot u_{se} = D_e \qquad \forall e \in \mathcal{E} \tag{2.1h}$$

$$(x_{se,t,er,r}, y_{se,t}, w_{se,er,r}, s_{c,p}) \in \mathcal{C}_G \tag{2.1i}$$

$$x_{se,t,er,r}, y_{se,t}, w_{se,er,r} \in \{0,1\} \tag{2.1j}$$

$$s_{c,p} \in \mathbb{N}_0 \tag{2.1k}$$

Constraint (2.1b) specifies that every event resource $er \in se$ of a sub-event $se \in \mathcal{SE}$ must be assigned one resource $r \in er$ (the resource can be the dummy-element). Constraint (2.1c) links variable $x_{se,t,er,r}$ to variable $y_{se,t}$. Together with Constraint (2.1b) these constraints ensure that only a single starting time is assigned to each sub-event. Constraint (2.1d) links variable $w_{se,er,r}$ to variable $x_{se,t,er,r}$. Constraints (2.1e) and (2.1f) determines whether a sub-event should be marked as active given the assigning of time and resources. Constraint (2.1g) ensures that a sub-event cannot be marked as active unless it fulfills the necessary criteria. Constraint (2.1h) makes sure that the sum of the duration of the active sub-events of an event equals the duration of the event. Constraint (2.1i) describes the fact that all native XHSTT constraints $\mathcal{C}_G$ can be handled using the variables of the model (and other auxiliary variables), such that variable $s_{c,p}$ takes the correct value for all constraints and their point-of-applications, slightly abusing notation.

Model (2.1) does not explicitly contains constraints such as those penalizing events which are not assigned the appropriate resources (*Assign Resource* constraints) and those penalizing conflicts between resources (*Avoid Clashes* constraints), since these are handled in the set $\mathcal{C}_G$. However, these constraints are part of all encountered XHSTT instances (in case of the *Assign Resource* constraints, these only apply to events which contain free event resources), and are natural constraints for any XHSTT instance. Furthermore they are essential to the described TSD of Section 2.3, so therefore both of these constraints are formalized in the following.

An *Assign Resource* constraint penalizes event resources of an event $e \in \mathcal{E}$ which is not assigned a resource. Let $\mathcal{C}_G^{\text{assignres}} \subseteq \mathcal{C}_G$ denote the set of *Assign Resource* constraints. By $e \in c$ it is denoted that constraint $c \in \mathcal{C}_G^{\text{assignres}}$ applies to event $e \in \mathcal{E}$. An *Assign Resource* constraint

contains a property denoted *role*, which must match the corresponding property of the applicable event resources of the events. Let $s^{\text{assignres}}_{c,er} \in \mathbb{N}_0$ be the slack of event resource $er \in e$ (which is a point-of-application) of event $e \in c$ in constraint $c \in \mathcal{C}^{\text{assignres}}_G$. Eq. (2.2) constraints this variable properly.

$$D_e - \sum_{se \in e, r \in er \setminus r_D} D_{se} w_{se,er,r} = s^{\text{assignres}}_{c,er} \qquad \forall c \in \mathcal{C}^{\text{assignres}}_G, e \in c, er \in e, \text{role}_c = \text{role}_{er} \tag{2.2}$$

Let $\mathcal{C}^{\text{avoidclashes}}_G \subseteq \mathcal{C}_G$ denote the set of *Avoid Clashes* constraints. Denote by $r \in c \in \mathcal{C}^{\text{avoidclashes}}_G$ that constraint c applies to resource r. Let the set $T^{\text{start}}_{se,t} \subseteq \mathcal{T}$ be the set of times which sub-event $se \in \mathcal{SE}$ lies in if it is assigned starting time $t \in \mathcal{T}$ (which is derived on the basis of the duration of the sub-event). The penalty value of an *Avoid Clashes* constraint for resource $r \in c \in \mathcal{C}^{\text{avoidclashes}}_G$ and time $t \in \mathcal{T}$ is denoted $s^{\text{avoidclashes}}_{c,r,t} \in \mathbb{N}_0$, and takes value $n-1$ iff resource r is used n times in time t and $n \geq 2$. Eq. (2.3) ensures this.

$$\sum_{se \in \mathcal{SE}, er \in se, t' \in T^{\text{start}}_{se,t}} x_{se,t',er,r} - 1 \leq s^{\text{avoidclashes}}_{c,r,t} \qquad \forall c \in \mathcal{C}^{\text{avoidclashes}}_G, r \in c, t \in \mathcal{T} \setminus t_D \tag{2.3}$$

2.2 The Danish High School Timetabling Problem

This section describes the DHSTP in detail. The shown model of the problem has been used in practice by many high schools in Denmark, and efficient solution approaches are an important tool for the quality of the annual timetable. It will be shown how the DHSTP relates to the GHSTP by similarity of the corresponding MIP models. However, this will show that the DHSTP is not fully contained in GHSTP, since a few aspects are currently not supported by the XHSTT format. Although, it should be said that the models share a lot of properties, and the DHSTP resembles the GHSTP in many aspects. An advantage of treating the DHSTP separately from the GHSTP is that specialized MIP models and solution approaches can be developed.

By definition, an instance of DHSTP has the following properties:

- All events have duration 1 (meaning that they span only a single time-slot), therefore only a single sub-event is necessary for each event.

- The resource requirements of an event consist of a known set of students, a known set of teachers, and a requirement for a single room. The set of *entities* $\mathcal{A}$ contains all students and teachers. Let the parameter $E_a \subseteq \mathcal{E}$ be the set of events which require entity $a \in \mathcal{A}$. Let the set of rooms be denoted $\mathcal{R}_D$. Each event requires a room to be assigned. Since all resources, except the room, for an event are known, the event resource index can be dropped from the basic decision variable, such that this is written as $x_{e,t,r}$, which takes value 1 if event $e \in \mathcal{E}$ is assigned time $t \in \mathcal{T}$ and room $r \in \mathcal{R}_D$, and 0 otherwise.

- No violation of hard constraints is allowed, so only the soft-constraints contribute to the value of the objective function. The set of hard constraints is denoted C^{hard}_D, and the set of soft-constraints is denoted C^{soft}_D.

- All constraints use the CostFunction `Sum`. Therefore the p-index can be dropped from from variable $s_{c,p}$.

Model (2.4) shows the MIP model of the DHSTP.

DHSTP MIP model (2.4)

$$\min \quad \sum_{c \in C_D^{\text{soft}}} s_c \tag{2.4a}$$

s.t.

$$\sum_{t \in \mathcal{T}, r \in \mathcal{R}_D} x_{e,t,r} = 1 \ \forall e \in \mathcal{E} \tag{2.4b}$$

$$\sum_{e \in \mathcal{E}} x_{e,t,r} \leq 1 \ \forall t \in \mathcal{T} \setminus t_D, r \in \mathcal{R}_D \setminus r_D \tag{2.4c}$$

$$\sum_{e \in E_a, r \in \mathcal{R}_D} x_{e,t,r} \leq 1 \ \forall t \in \mathcal{T} \setminus t_D, a \in \mathcal{A} \tag{2.4d}$$

$$(x_{e,t,r}, s_c) \in C_D^{\text{soft}} \tag{2.4e}$$

$$x_{e,t,r} \in C_D^{\text{hard}} \tag{2.4f}$$

$$x_{e,t,r} \in \{0,1\} \tag{2.4g}$$

$$s_c \in \mathbb{N}_0 \tag{2.4h}$$

Constraint (2.4b) specifies that each event should be assigned one time and one room. Constraints (2.4c) and (2.4d) specify no conflicts among rooms and resources, respectively. Constraints (2.4e) and (2.4f) ensures that the soft constraints C_D^{soft} should be penalized accordingly in the objective, and that the hard constraints C_D^{hard} should be respected, respectively.

Even though the DHSTP is not fully contained in GHSTP, a conversion from a DHSTP-instance to a GHSTP-instance is interesting in terms of making approximated versions of DHSTP instances available in the XHSTT format. Paper J contains a full conversion scheme from DHSTP to GHSTP. This scheme illustrates that due to limitations of the XHSTT format, the DHSTP cannot be modeled completely as-is. However, most of these limitations are minor flaws, and the resulting XHSTT instances resemble the DHSTP in the major aspects. The effect of the critical flaws is that the XHSTT instances might contain inevitable violations of some hard constraints. The conversion scheme is briefly described below.

- The set of events $\mathcal{E}$ and the set of times $\mathcal{T}$ are analogous.
- The set of resources $\mathcal{R}$ consists of all entities and rooms, $\mathcal{R} = \{\mathcal{A} \cup \mathcal{R}_D\}$.
- Since all events have duration 1, all sub-events should be active, so variable u_{se} can be dropped as well as eqs. (2.1e) to (2.1h). Thereby Constraint (2.4b) resembles constraints (2.1b), (2.1c), and (2.2).
- For each event $e \in \mathcal{E}$ and for each entity $\{a \in \mathcal{A} \mid e \in E_a\}$, an event resource of the GHSTP event is created which is fixed to the resource representing the entity.
- For each event $e \in \mathcal{E}$, a free event resource is created for the GHSTP event which requires the assignment of a room $r \in \mathcal{R}_D$.
- Constraints (2.4c) and (2.4d) are handled by creating a single *Avoid Clashes* constraint (Constraint (2.3)) which is marked as a hard constraint.

What remains to be shown is that constraints C_D^{soft} and C_D^{hard} are contained in constraints $\mathcal{C}_G$. This is beyond the scope of this chapter, and the reader should look into Paper J. As already discussed, this illustrates certain limitations of the XHSTT format.

2.3 Two-Stage Decomposition

A recent decomposition method is *Two-Stage Decomposition* (TSD), which is a promising branch of exact methods for timetabling problems. The theory of TSD is closely tied with the optimization problem in question, so therefore the TSD is presented on the basis of the description of the GHSTP (see also Paper D). However, a description of TSD for the DHSTP would be analogous (see Paper C).

TSD was first applied to the Curriculum-based University Timetabling problem of the International Timetabling Competition 2007, see Lach and Lübbecke (2008, 2012), with good results. The subsequent papers of Hao and Benlic (2011) and Cacchiani et al. (2013) show how the TSD lies as a foundation for more advanced solution approaches for this problem.

The basic idea of TSD is to split the problem in two stages: In the first stage (*Stage I*), events are assigned to times subject to *anonymous assignment* of resources to events. By this is meant that the assignment of events to times is performed such it is ensured that an assignment of resources can be performed subsequently, but it is currently not decided which specific resources that are assigned to each event. The actual assignment of resources is performed in *Stage II*, where the allocation to times provided by Stage I is considered as fixed. This means that Stage I is formulated such that a solution is also feasible (or even optimal) in terms of Stage II. The approach and its relation to the original MIP model are illustrated on Figure 2.1 for the GHSTP. It is important to notice that Stage I and Stage II are solved in sequence, i.e. the TSD is not an iterative approach. The only native XHSTT constraints described in Section 2.1 are *Assign Resource* and *Avoid Clashes*, and these are handled in the basic version of TSD described here. For a TSD considering all native XHSTT constraints, see Paper D.

Stage I is defined in terms of variable $y_{se,t}$, and Stage II is defined in terms of variable $w_{se,er,r}$. The anonymous assignment of resources required in Stage I is carried out by a priori building a bipartite graph for each time $t \in \mathcal{T}$, denoted G_t. Denote by A the set of all applicable sub-events and event resource combinations, i.e. $A = \{ (se, er) \mid se \in \mathcal{SE}, er \in se \}$. The graph contains a vertex for every element in A and for every resource, and the set of edges is denoted E_t, i.e. $G_t = \{A \cup \mathcal{R}, E_t\}$. Iff a sub-event can be assigned to a resource; an edge exists between each element of A which contains the sub-event and the resource. The graph contains a t-index, as the set of edges (and their weights) might differ between times due to certain constraints. However, in the context of constraints *Assign Resource* and *Avoid Clashes* this is not the case, but the t-index is kept for generality.

The existence of a *matching* in the bipartite graph G_t determines feasibility of the resource assignment for time $t \in \mathcal{T}$. To this end the famous *Hall's Marriage Theorem* is imposed. Denote by $\Gamma(S) \subseteq \mathcal{R}$ the neighbors of $S \subseteq A$ in graph G, i.e. $\Gamma(S) = \{ r \mid (a, r) \in E, a \in S, r \in \mathcal{R} \}$.

Theorem 1 (Hall's Theorem)**.** *The bipartite graph $G = (A \cup \mathcal{R}, E)$ has a matching of all vertices A into $\mathcal{R}$ if and only if $|\Gamma(S)| \geq |S| \ \ \forall S \subseteq A$.*

The implication of this theorem is that for a given time $t \in \mathcal{T}$ the feasibility of the matching can be determined by the amount of elements of A assigned to time t. Denote by $se \in A$ the elements of A which se are part of. For a given subset $S \subseteq A$, the neighbors $\Gamma(S)$ can be determined a priori (because the graph is constructed initially). Furthermore, the cardinality of S can be substituted by $\sum_{se \in S, t' \in T^{\text{start}}_{se,t}} y_{se,t'}$. Thereby the feasibility of the resource matching of Stage II can be determined in Stage I, maintaining optimality of the TSD. Specifically, the following constraint are introduced in Stage I,

$$\sum_{se \in S, t' \in T^{\text{start}}_{se,t}} y_{se,t'} \leq |\Gamma(S)| \qquad \forall S \subseteq A_t, t \in \mathcal{T} \tag{2.5}$$

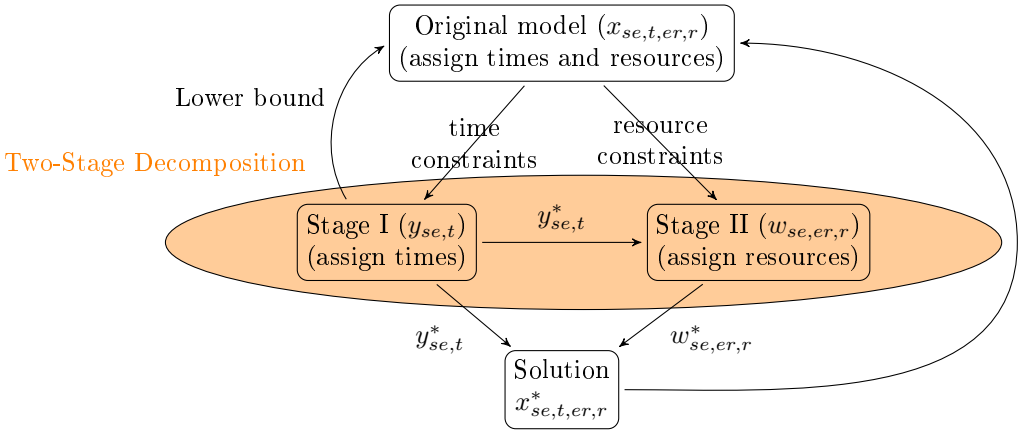


Figure 2.1: Outline of Two-Stage Decomposition.

If a solution to Stage I is feasible w.r.t. (2.5), then it is ensured that the violation of constraints *Assign Resource* and *Avoid Clashes* is zero. However, since the GHSTP allows violating all constraints (including hard constraints), further steps are required (which are beyond the scope of this section to describe).

Eq. (2.5) describes an exponential amount of inequalities, so not all of these can be added to the Stage I MIP model. Instead the only inequalities which are added are those necessary for the specific problem instance. This means that inequalities which are dominated by other inequalities are left out. In practice for the GHSTP (and for the DHSTP), it is shown that only a small tractable subset of these inequalities are required. Notice that these inequalities can be generated upfront (since only few of them are needed), contrary to generating them on-the-fly as the algorithm proceeds.

This concludes the basic description of the TSD. Paper C and Paper D provide more details. To summarize, the advantages of TSD are:

- The total number of variables in the model is significantly reduced (e.g. in the context of GHSTP, variable $x_{se,t,er,r}$ is dropped altogether).
- Instead of solving the entire model at once, the model is split into two smaller models which are solved sequentially.
- For both the GHSTP and the DHSTP, the majority of constraints can be handled optimally. However, this suggests that some constraints cannot be handled by the decomposition, but it is shown in practice that this has little impact for both the GHSTP and the DHSTP.

As for the disadvantages, the TSD is still tied closely with its application (the optimization problem). Since the present theory does not contain a sufficient abstraction of the method, it is difficult to suggest other optimization problems where it can be applied. However, TSD does require a certain structure in a problem to be applied, so it is not a general approach for MIP models.

2.4 International Timetabling Competition 2011

The *International Timetabling Competition 2011* (ITC2011) ran during 2011-2012 (Post et al., 2013). The datasets of the competition were based on archives containing a number of XHSTT

instances, specifically the archives `XHSTT-2012` and `XHSTT-ITC2011-hidden`. The competition was preceded by the competitions Paechter et al. (2002) and McCollum et al. (2010), which had great impact on the timetabling community. The overall objectives of the competition were the following (Post, 2013a):

- Allow researchers to trial their techniques in a competitive setting on 'real world' practical problems.
- Encourage research in the area of complex $\mathcal{NP}$-hard real world problems.
- Attract researchers from all disciplines to compete.
- Further algorithmic development in the area of educational development.
- Generate all-time best solutions to these problem instances.

These objectives illustrate that having the XHSTT format as a common ground for researchers is a large advantage for the high school timetabling community. A total of 17 teams participated in the competition, and the finalist teams of Round 2 were the following:

- GOAL: This team of University of Ouro Preto used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al., 2012).
- HySTT: Participant from the University of Nottingham used a method based on Hyper-heuristics (Kheiri et al., 2012).
- Lectio: This team of Technical University of Denmark used an algorithm based on ALNS, see Paper I.
- HFT: From University of Applied Sciences Stuttgart, this team applied an Evolutionary Algorithm (Romrös and Homberger, 2012).
- VAGOS: The algorithm of this team is so far not documented.

Table 2.2 summarizes the results of Round 2 and Round 3 of the competition. In Round 1 of the competition, participants were allowed to submit solutions to several known instances without any restrictions on the computational method. In Round 2, the algorithms of the finalist teams were tested on 18 hidden (previously unseen) instances. The algorithms were run 10 times on each instance on the same machine given the same time limit and different random seeds. Based on these runs, the teams were ranked on their average performance. Round 3 had the same rules as Round 2 and involved the same instances, but with unlimited computational restrictions. In both rounds the ALNS algorithm of team Lectio finished third among the finalists.

Table 2.2: ITC2011 results (Post et al., 2013). For each participating team and each round is shown the average rank obtained across all instances.

	GOAL	HySTT	Lectio	HFT	VAGOS
ITC2011 Round 2	1.18	2.23	2.32	3.64	-
ITC2011 Round 3	1.64	2.25	2.75	3.75	3.86

Chapter 3

The Generalized Meeting Planning Problem

The *Generalized Meeting Planning Problem* (GMPP) is a generalization of the following optimization problem: Given is a set of entities $\mathcal{A}$, a set of meetings $\mathcal{G}$, a parameter $C_{a,g} \in \{0,1\}$ which takes value 1 if entity $a \in \mathcal{A}$ is part of meeting $g \in \mathcal{G}$, and a set of time-slots $\mathcal{T}$. The goal is to maximize the number of meetings assigned to a time-slot, such that no clashes among entities occur, subject to other problem specific soft- and hard-constraints. In this section the GMPP is set up as a MIP model, and its relation to the GHSTP is discussed. The possible applications of the GMPP are many, and some examples from the educational sector are described in the following:

- In the context of the HSTP, a meeting corresponds to a lecture and entities correspond to resources. The set of time-slots is analogous. This suggests that high school timetabling problems can be solved by the GMPP.
- In the University Course Timetabling Problem a number of lectures for courses should be scheduled in a weekly timetable. This problem can be handled in the same way as HSTP in the GMPP, where lectures are defined in terms of meetings.
- For the Examination Timetabling Problem, the goal is to schedule exams to times such that the schedule for students and teacher fulfill various criteria. Thereby an exam corresponds to a meeting in the GMPP, and students and teachers constitute the set of entities.

To formalize a MIP model of the GMPP, the variable $x_{g,t} \in \{0,1\}$ is introduced which takes value 1 if meeting $g \in \mathcal{G}$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. The parameter $\alpha_{g,t} \in \mathbb{R}^+$ denotes the profit of scheduling meeting $g \in \mathcal{G}$ to time $t \in \mathcal{T}$. The solution procedure of this model is based on a *Column Generation* (CG) procedure, which is used to solve (large) LP problems. A CG procedure consists of a *master problem*, which is analogous (possibly a reformulation) to the original problem, and a *subproblem*. Typically, the formulation of the master problem contains too many variables (columns) to be generated upfront. Instead the *restricted master problem* is considered, which contains only a subset of columns. The CG procedure proceeds as follows: The restricted master problem is solved using the current set of columns, obtaining the dual prices of the constraints. Given these dual values, the subproblem is able to identify new columns. The objective of the subproblem is to minimize the reduced cost (assuming minimization of the original problem) of the new column w.r.t. the dual variables and

the naturally occurring constraints. If the objective of the optimal solution of the subproblem is negative, the column enters the restricted master problem, and the procedure repeats (i.e. the restricted master problem is solved next). If the objective of the subproblem is non-negative, the optimal solution to the master problem has been found.

In context of MIP models, CG does not work out of the box. Instead a *Branch-and-Price* (B&P) algorithm can be used. In a B&P algorithm, the LP-relaxation of a MIP model is considered (usually in a reformulated way, for instance by applying Dantzig-Wolfe Decomposition) and solved by CG. This is embedded in a Branch-and-Bound framework to ensure integrality of the solution. See Barnhart et al. (1998) for a throughout description of B&P algorithms. Paper G describes the B&P algorithm for the GMPP in detail.

To formalize the GMPP in context of a B&P algorithm the set of *entity patterns* $\mathcal{P}$ is introduced. An entity pattern is an *anonymous schedule* for an entity, which contains information on which times the entity is busy, but does not contain information on the specific meetings the entity attends. The number of patterns for each entity might be exponential, but for now it is assumed that all of them are known. The set $P_a \subseteq \mathcal{P}$ denotes the set of patterns for entity $a \in \mathcal{A}$. Let parameter $M_{a,p,t} \in \{0,1\}$ take value 1 if pattern $p \in \mathcal{P}$ belongs to entity $a \in \mathcal{A}$ and the pattern *does not* allow a meeting in time $t \in \mathcal{T}$, and 0 otherwise. The parameter $\beta_{a,p} \in \mathbb{R}$ denotes the profit (possibly negative) of using entity pattern $p \in \mathcal{P}$ which belongs to entity $a \in \mathcal{A}$. Let variable $\lambda_{a,p} \in \{0,1\}$ take value 1 if entity $a \in \mathcal{A}$ is using the schedule of entity pattern $p \in \mathcal{P}$, and 0 otherwise.

This CG formulation of the GMPP has the advantage of encapsulating problem-specific constraints in the sub-problem. This makes it more widely applicable.

Model (3.1) shows the master problem of the GMPP.

GMPP MIP Master Problem (3.1)

$$\max \sum_{g\in\mathcal{G},t\in\mathcal{T}} \alpha_{g,t}x_{g,t} + \sum_{a\in\mathcal{A},p\in P_a} \beta_{a,p}\lambda_{a,p} \tag{3.1a}$$

s.t.

$$\sum_{t\in\mathcal{T}} x_{g,t} \leq 1 \ \forall g \in \mathcal{G} \tag{3.1b}$$

$$\sum_{g\in\mathcal{G}} C_{a,g}x_{g,t} + \sum_{p\in P_a} M_{a,p,t}\lambda_{a,p} = 1 \ \forall a \in \mathcal{A}, t \in \mathcal{T} \tag{3.1c}$$

$$\sum_{p\in P_a} \lambda_{a,p} = 1 \ \forall a \in \mathcal{A} \tag{3.1d}$$

$$x_{g,t}, \lambda_{a,p} \in \{0,1\} \tag{3.1e}$$

Constraint (3.1b) ensures that each meeting is not assigned to more than one time. Constraint (3.1c) relates variables $x_{g,t}$ and $\lambda_{a,p}$, and specifies that if a meeting is assigned a time, then all entities of the meeting must be assigned a pattern that allows them to have a meeting in this time. Furthermore this constraint ensures that an entity is assigned at most one meeting in each time. Constraint (3.1d) ensures that each entity is only assigned one pattern.

For relating GMPP to the GHSTP, it is now attempted to transform an instance of GMPP into an instance of GHSTP. First of all, an event with duration 1 is created for each meeting $g \in \mathcal{G}$. Thereby the set of entities $\mathcal{A}$ and the set of times $\mathcal{T}$ of the GMPP instance equals the set of resources $\mathcal{R}$ and the set of times $\mathcal{T}$ of the GHSTP instance, respectively. An *Avoid Clashes* constraint is required to model constraint (3.1c). Thereby Model (3.1) is contained in the GHSTP instance; however the subproblem remains to be dealt with. Clearly, if all problem-specific

constraints modeled by the subproblem of the GMPP instance can be analogously formulated in GHSTP, any instance of GMPP can be solved by the GHSTP model. In fact, for the considered use case of the GMPP described in the next section, the problem could be solved by the GHSTP model.

3.1 The Consultation Timetabling Problem

In this section the *Consultation Timetabling Problem* (CTP) is described, and its relationship with the GHSTP is discussed.

The CTP considers the scheduling of meetings between students and teachers to times. The times might be spread across multiple days (usually the time horizon is short, e.g. below five days). The primary aim of the meeting is to allow the teachers to give individual feedback to the students. The participants of each meeting are usually only a single student and one or two teachers. The goal of the CTP is to schedule the meetings such that the meeting plan for each student and teacher is as desirable as possible, elaborated later in this section.

Two versions of the CTP are considered, the *Parental Consultation Timetabling Problem* (PCTP) and the *Supervisor Consultation Timetabling Problem* (SCTP). In the PCTP, the students are usually joined by their parents for the meeting. The PCTP is usually held in the evening of work days. In the SCTP, the teachers usually have the role of supervisors for the student for a specific project of the teaching curriculum. The SCTP is held during normal school hours. Paper F shows a uniform model of the CTP, which handles both the PCTP and the SCTP. In this model, the difference between the PCTP and the SCTP lies in differences in certain parameter values, most of them defining different preferences w.r.t. solution characteristics. Further details are beyond this section to describe (see Paper F for more information). The constraints of the model of the CTP are similar for the PCTP and the SCTP, and these are described in the following, along with the relation to the GHSTP:

- Students or teachers can be occupied by other activities at certain times. This is handled using *Avoid Unavailable Times* constraints.
- An idle time-slot for a student/teacher is defined as a time-slot with no meetings scheduled, but there exists both an earlier and a later time-slot where a meeting is scheduled. Idle time-slots are undesirable for both students and teachers, and constitute a penalty in the objective. This corresponds to *Limit Idle Times* constraints of the GHSTP.
- It is undesirable for both students and teachers to have too long sequences of meetings without breaks. This is handled by a *Limit Busy Times* constraint.
- If the times are spread across multiple days, it is undesirable for both students and teachers to be assigned times on more than one day. Specifically, the number of 'excess' days is penalized in the objective. This is handled in the GHSTP by *Cluster Busy Times* constraints.
- In the weight $\alpha_{g,t}$ is contained a penalty for assigning of meetings to certain times (independent of the entities which are part of the meeting). This can be handled using *Prefer Times* constraints.

Thereby it has been argued that it is possible to convert an instance of CTP to the GHSTP.

The structure of the CTP fits into the GMPP model, so the CTP can be considered as a special-case of the GMPP. Notice that all constraints of the CTP can be handled in the subproblem of the GMPP. Details of this application can be found in Paper G.

Chapter 4

Overview of Results

This chapter describes the scientific contributions of the thesis, as well as the practical applications of the developed solution algorithms. Throughout this chapter the average worst-case gap to optimality is used as an indicator on how well an algorithm performs for a given optimization problem. The gap between a solution s and a bound $\overline{s}$ equals $100\frac{|s-\overline{s}|}{s}$ for each dataset, and the average gap is the average of these gaps taken over a set of datasets. However, the performance of an algorithm should also be evaluated based on other criteria, such as the given resource-limitations (e.g. the imposed time-limit). Furthermore it is not guaranteed that a solution method finds a feasible solution, nor can it be guaranteed that a bound is known for a dataset. In such cases the gap cannot be calculated. Therefore the comparison of the average gap between different solution methods might be based on approximate numbers, but nevertheless these average gaps are considered as important measures of performance in the following.

4.1 Papers and Conferences

This thesis is composed of seven scientific papers, which are either published or submitted to peer-reviewed journals. In this section the content of these papers are described. The papers can be seen in their full length in Part II. Furthermore, additional papers are listed in the appendix Part III, and these are also briefly described in this section.

4.1.1 Paper A: Integer Programming for the Generalized (High) School Timetabling Problem

Submitted to Journal of Scheduling, Sep. 2013

This paper describes a MIP model for the XHSTT format, which is capable of handling all types of native XHSTT constraints, and is thereby applicable to any given instance. This constitutes the first exact method for XHSTT. Instead of solving the MIP model directly using a generic MIP solver, an approach is suggested which takes advantage of the objective structure of XHSTT. Specifically, this is done using the *lexicographic* method for multi-objective problems (Ehrgott, 2000). Besides generating incumbent solutions, this solution method is theoretically capable of finding lower bounds on optimum, which can lead to optimal solutions. The model is applied to most instances currently available in the XHSTT format and is in practice capable of producing 2 new optimal solutions, prove optimality of 4 known solutions, and generate 9 new sub-optimal

solutions. The new state-of-the-art solutions have been made available on the XHSTT website (Post, 2013b).

4.1.2 Paper B: Integer Programming and Adaptive Large Neighborhood Search for Real-World Instances of High School Timetabling

Submitted to Annals of Operations Research Jan. 2013, revision submitted Oct. 2013

This paper describes a model for the DHSTP, which is *complete* in the sense that it includes all constraints also considered in practical applications by the high school administration. A MIP model is also developed. Three different solutions algorithms are proposed: Solving the MIP using a state-of-the-art MIP solver, a basic version of TSD, and a heuristic based on the ALNS paradigm. The algorithms are tested on 100 real-world datasets, and computational results show that the ALNS heuristic performs best. The gap between the found solution and the best available bound is in average 65.3%, 41.3% and 25.6% for the MIP model, the TSD and the ALNS heuristic, respectively, which are all rather high gaps. Three of the datasets are converted to the XHSTT format, and made publicly available.

4.1.3 Paper C: A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark

Published in Computers & Operations Research, vol. 43, pp. 36-49, March 2014
This work was presented at ECCO 2012.

This paper extends the basic version of TSD used in Paper B for the DHSTP, such that it now incorporates edge weights in the anonymous resource assignment graph. This makes for a more efficient decomposition in theory, and computational results indeed show that the generated bounds for the 100 tested instances in general are better than the ones of the basic TSD. However, the additional constraints have a negative effect in terms of solution quality, and the extended method is outperformed by the basic TSD. Given these improved solutions and bounds, it is shown that the gap between the best known solution and the best known bound is 22.3% in average for the DHSTP. The paper also includes a detailed outline of a problem-specific way to generate the necessary Hall's inequality. This approach is sufficient for 98 of the 100 considered instances. For the remaining two instances, the algorithm is still functional, but it cannot be guaranteed that all necessary Hall inequalities are generated, which has a possible bad effect on the quality of the solution and the generated lower bounds. Nevertheless, the paper shows that TSD is a promising MIP-based solution approach for the DHSTP.

4.1.4 Paper D: Decomposing the Generalized High School Timetabling Problem

Submitted to INFORMS Journal on Computing, Nov. 2013

Based on the MIP model of Paper A, this manuscript describes in detail how the MIP model of the GHSTP can be decomposed by the TSD method, thereby reducing the total amount of variables considerably. Optimality of the original MIP model is maintained in the process, except for some datasets which have unfortunate characteristics. These characteristics are a result of the large amount of different constraints in the XHSTT format. However, for most of these unfortunate cases, the algorithm is still capable of producing lower bounds on optimum. This

TSD is a first-step towards more advanced exact methods for the GHSTP. Computational results are obtained using 12 XHSTT instances which are among the largest instances available, and show that the models yielded by the TSD are of much smaller size than those of the original MIP model. However, the quality of the found solutions and lower bounds are not as high as expected for the tested XHSTT instances. It is argued that the TSD is an advantage for these instances, but also inadequate for establishing state-of-the-art results. Further enhancements are required before this exact method can compete with the heuristics for the GHSTP.

4.1.5 Paper E: A Matheuristic for High School Timetabling

Submitted to European Journal of Operational Research, Nov. 2013

A matheuristic is described which is a hybridization of integer programming and metaheuristics. By construction, the heuristic uses problem-specific knowledge to divide the solution space into different parts, and solves each part w.r.t. the complete MIP model. Different parts of the solution space are considered as the algorithm progresses, constituting different neighborhoods of the current solution. Apart from the problem-specific knowledge used to construct neighborhoods, the matheuristic is generally applicable to MIP models. An adaptive layer of the algorithm uses feedback from the solution process to select the type of neighborhoods which have performed best so far, and favors these neighborhoods in future selections. The algorithm is applied to both the GHSTP and the DHSTP, and computational results show state-of-the-art performance for both problems. For the GHSTP, the performance of the algorithm is comparable with the winner of Round 2 of ITC2011. For the DHSTP, the matheuristic is the best algorithm compared to three others given a time-limit of two hours. In fact, the solutions provided by the matheuristic narrowed the gap from best known solution to best known bound to 15.2% in average for the DHSTP, given a two hour time-limit. Given a four minute time-limit, the average gap is 23.4%.

4.1.6 Paper F: The Consultation Timetabling Problem at Danish High Schools

Published in Journal of Heuristics, vol. 19, 3, pp. 465-495, Jun. 2013

This paper describes the CTP for the first time in the literature, including a motivation and detailed description of each constraint, and the development of a MIP model. This MIP model applies to both the PCTP and the SCTP, and is shown to be a challenge for the commercial MIP-solver Gurobi 5.0.1. A proof of the model being $\mathcal{NP}$-hard is given by reduction from Graph Coloring. Computational results are established using 300 real-life datasets, and an ALNS algorithm is developed, which is shown to generally perform better than Gurobi. The ALNS algorithm is also shown to outperform another heuristic for the problem which is used by high schools in practice. Using the bounds obtained with Gurobi, it is shown that the ALNS heuristic produces solutions which are within 5% of optimum in average.

4.1.7 Paper G: A Branch & Price Algorithm for the Generalized Meeting Planning Problem

Submitted to Computers & Operations Research, Aug. 2013

This paper describes the GMPP in detail, which is a generalization of the CTP of Paper F. The model of the GMPP consists of a master problem of a CG procedure, constructed such that the problem-specific details are handled by the subproblem. This makes the model capable of handling a variety of timetabling problems. This CG procedure is embedded in a B&P framework,

which uses *Strong Branching* to speed up the algorithm. Computational results reveal two things for the CTP: 1) The B&P algorithm performs very well 2) New bounds are provided which show that the ALNS algorithm is within 2.31% and 1.26% of optimum for the PCTP and the SCTP, respectively.

4.1.8 Other Papers

The following papers, technical reports and conference abstracts were also produced during the Ph.D. study.

- Paper H: Elective Course Planning
  Published in European Journal of Operations Research, vol. 215, 3, pp. 713-720, Dec. 2011

  This paper considers the *Elective Course Planning Problem* at Danish high schools, which is the first description of this problem in the literature. A MIP model of the problem is presented, which is decomposed into a CG model. This CG model is solved in a B&P framework, where the subproblem is handled by a polynomial algorithm. The algorithm is tested on 98 real-life datasets, with promising results. It is argued that the algorithm outperforms an existing metaheuristic. To enhance the B&P algorithm, *Explicit Constraint Branching* is used.

- Paper I: International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm
  Presented at the Ninth International Conference on the Practice and Theory of Automated Timetabling (2012)

  This text describes a contribution to ITC2011, namely an ALNS algorithm. This algorithm was part of the final rounds of ITC2011, and received a third place in both Round 2 and Round 3 of the competition. Furthermore, the algorithm produced several new best solutions for different XHSTT instances. See also Section 2.4.

- Paper J: Comparing Solution Approaches for a Complete Model of High School Timetabling
  Technical Report 5.2013, DTU Management Engineering, Technical University of Denmark

  This technical report largely contains the same material as Paper B, and is only partially included in this thesis. Extra material covered is the formal proof of the DHSTP being $\mathcal{NP}$-hard and a scheme for converting a DHSTP instance to an instance of the XHSTT format.

- High School Timetabling: Modeling and solving a large number of cases in Denmark
  Presented at the Ninth International Conference on the Practice and Theory of Automated Timetabling (2012)

  This conference text contains preliminary models and results of Paper B. It was presented at PATAT2012. The text is not included in this thesis.

4.2 Scientific Contributions

The scientific contributions of this thesis are listed in the following.

- The first MIP model of the XHSTT format, and thereby the first exact method, has been presented. This model handles an arbitrary instance of XHSTT, and has been applied

to the most recent version of all XHSTT datasets. This has yielded the first proofs of optimality for some of these benchmark instances, as well as providing new lower bounds for other instances.

- A throughout model of the Danish case of DHSTP has been shown. This includes all constraints required by the high schools in a practical setting, such that the model specifications are *complete* in this sense. This is the first time this problem has been described in the scientific literature. Furthermore, a MIP model has been presented. Three datasets have been made available in the XHSTT format, which are part of the archive `XHSTT-2013`. The MIP has been evaluated on 100 real-world instances.

- Heuristics based on ALNS have been developed for the considered optimization problems. The application of ALNS to timetabling problems has not been described before in the literature.

  - For the DHSTP, an ALNS algorithm is currently among the best known algorithms. The average gap to the currently best known bounds from the solutions obtained is 19.5% given a four minute time-limit.
  - For the GHSTP, an ALNS algorithm was among the finalists in ITC2011, and is among the best heuristics currently available.
  - For 200 instances of the CTP an ALNS algorithm showed to perform within 3% of optimum.

- A matheuristic applicable to both DHSTP and GHSTP has been created, and has shown good results for both problems. For the GHSTP, the algorithm was shown to be competitive with the winner of Round 2 of ITC2011. For the DHSTP, the algorithm outperforms all considered MIP approaches, and comes close to the performance of the ALNS algorithm. Given a two hour time-limit, the average gap from the solutions of the matheuristic to the best known bound is 15.2% on 100 instances of the DHSTP. The adaptive layer of the algorithm resembles that of an ALNS algorithm, and the matheuristic can be thought of as a hybrid between MIP and ALNS.

- An extension of TSD has been developed, which makes the technique applicable to a wider range of problems. The extension considers edge-weights in the graph which models the anonymous assignment of resources, and shows how a lower bound on this weighted assignment can be derived. This extension has shown to be required for handling both the DHSTP and the GHSTP using TSD. A problem-specific way of generating the necessary Hall inequalities has been developed for the DHSTP, and the same approach has shown to also be applicable for the GHSTP.

- The CTP has been described in detail. Like for the HSTP, the problem specifications of the CTP resemble all constraints necessary to handle a practical problem instance. This is the first description of the CTP in the literature. A MIP model has been developed, and an ALNS algorithm suggested.

- For the GMPP, a model has been described, as well as a discussion of the applicability to a range of problems within the educational sector. The GMPP is solved using a B&P approach, which is shown to be effective on 200 test-instances of the CTP problem.

4.3 Practical Applications

Practical usability is relevant for this Ph.D. study as it was done under the *Industrial Ph.D. Programme*. Some of the solution algorithms developed throughout the study have been made commercially available to end-users, and is used by a number of high schools. Thereby the models of the optimization problems are tested in a practical setting, which ensures that all necessary requirements are met. In this context it should be mentioned that the considered problem instances of the GHSTP, the DHSTP and the CTP reflect real-life optimization problems.

Throughout the Ph.D. project, the take on the considered optimization problems has in general been the following: Consider first a mathematical model (in practice, MIP models), and attempt to solve this using standard techniques (in practice, generic MIP-solvers). If this is successful within the given boundaries (i.e. time- and resource-limitations), then optimality has been achieved and no further work is required. In case of inadequate results (all timetabling problems considered in this thesis fall under this category), more specialized algorithms need to be designed.

In the following the current practical applications of the contributed algorithms are described for each optimization problem.

- HSTP: The ALNS algorithm of Paper B was made available to users of the timetabling component of Lectio on the 27th of February 2012. This algorithm has shown to be the one performing best among all considered solution approaches. Many high schools in Denmark use this algorithm in support of creating their yearly (or half-yearly) timetable.
- CTP: Also the ALNS algorithm of Paper F has gone into production, and is used by many high schools for producing timetables for the CTP.
- GHSTP: The developed algorithms for this problem have not been made available to end-users (this is not relevant for the users of Lectio). However in the context of practical applications, it should be mentioned that the instances of ITC2011 was classified as *unsimplified instances from real high schools around the world* in Post et al. (2013).

For these practical applications, it applies that the algorithms are used in a *decision-support context*. This means that the algorithms should be used as a helpful tool, capable of quickly providing sufficiently good solutions. The users of the system are given a certain amount of control of the solution process, e.g. by allowing them to adjust different weights w.r.t. properties of the desired solution. Furthermore, the users are able to edit the found solutions by appropriate graphic interfaces.

Chapter 5

Conclusion

This thesis contains contributions to the modeling and solution of difficult timetabling problems originating from practical applications at high schools. The considered timetabling problems are the Generalized High School Timetabling Problem (GHSTP), the Danish High School Timetabling Problem (DHSTP) and the Consultation Timetabling Problem (CTP). Furthermore the Generalized Meeting Planning Problem (GMPP) has been considered which is a framework for solving a range of timetabling problems.

Two important characteristics of these timetabling problems are: 1) They are hard to solve, and 2) The quality of the obtained solutions define the general satisfaction with the timetable for the stakeholders, i.e. good solutions are important. These timetabling problems are considered as optimization problems, and applying exact solution methods will lead to the optimal solution. Generally speaking, this thesis has advanced the fields of both heuristics and exact methods for real-world timetabling problems. Furthermore, contributions to the modeling of these problems w.r.t. Mixed-Integer Programming (MIP) models have been made.

In the high school timetabling problem, events require a set of resources and are sought scheduled to times, such that the timetable of each individual resource respects certain hard-constraints and minimizes the violation of the soft-constraints. For the GHSTP and the DHSTP, the contributions of this thesis can be summarized as follows.

- MIP models of these unsimplified high school timetabling problems have been described, which facilitates the use of exact solution methods. For the GHSTP, the solution of the corresponding MIP model has yielded new optimal solutions and lower bounds for standard benchmark instances.
- Heuristics based on Adaptive Large Neighborhood Search (ALNS) have been developed and shown to perform well for both problems.
- Novel extensions of the Two-Stage Decomposition (TSD) method allow a wider range of problems to be handled, and has shown good results for both the GHSTP and the DHSTP. This method has a large potential for future applications.
- A heuristic hybridizing mathematical programming and metaheuristics, known as a matheuristic, has shown good performance for both problems when compared to other solution approaches.
- Benchmark datasets representing three real-world timetabling instances of DHSTP have been made publicly available.

For all solution methods for the DHSTP, the average gap from the best known bounds to the obtained solutions is high when tested on 100 real-life instances with various time-limits. The current best known result is found by the matheuristic which obtains an average gap of 15.2% given a 2 hour time-limit. Section 5.1 discusses means for narrowing this gap.

This thesis has shown that real-world high school timetabling problems are still a challenge to solve for exact methods, even with the recent advances of generic MIP solvers and when applying state-of-the-art techniques such as TSD. In a practical setting for these problems, the tests performed in this thesis show that heuristics in general produce the best solutions. Both ALNS heuristics and the matheuristic have shown to perform well. However, exact methods which can provide bounds on optimum are valuable for evaluating the performance of the heuristics.

The CTP concerns the scheduling of meetings between students and teachers. Each meeting should be scheduled to a time, such that the individual timetables are as desirable as possible. Two versions of the CTP have been examined, the *Parental Consultation Timetabling Problem* and the *Supervisor Consultation Timetabling Problem*, which are important planning problems for the high schools in Denmark. A generalization of the CTP is the *Generalized Meeting Planning Problem* (GMPP) which has a wide range of applications. The contributions w.r.t. the CTP and the GMPP are summarized in the following.

- The CTP has been modeled and a MIP model established for the first time in the literature.
- A Column Generation scheme and a Branch-and-Price (B&P) algorithm have been developed for the GMPP. In case of the CTP, this Branch-and-Price algorithm produces solutions which are within 3% of optimum in average (based on 200 real-life datasets).
- An ALNS heuristic has been developed for the CTP, and this heuristic finds solutions which are within 3% of optimum in average.

Contrary to the case of HSTP, the performance of an exact algorithm (the B&P algorithm) for the CTP is competitive with the performance of the tested heuristic (the ALNS heuristic). This shows that it may be possible to use exact methods for CTP in a practical setting.

5.1 Future Research

Generally, the amount of literature concerned with exact methods for timetabling problems is still low compared to that of heuristics. The MIP models presented in this thesis can serve as a basis for more advanced exact methods in future research. The establishment of lower bounds or even optimal solutions is important for the high schools, but also for evaluating solution methods.

For determining the timetabling problems which are subject for future research, the magnitude of the average gap to the best known bounds is inspected. For the DHSTP, the current best average gap is 15.2% (given a two hour time-limit for each dataset). This is a significant gap, which future research should address. For the CTP, both the ALNS algorithm and the B&P algorithm achieve solutions which are within an average gap of 3%. This is a low gap, and thereby the CTP is not an obvious case for future research. In terms of the GHSTP, a bound is not known for a lot of datasets, so a sufficient basis for calculating the average gap does not exist. Providing (tight) bounds for all GHSTP instances is an obvious topic for future research.

For closing the gap to optimum, one can either seek for improved solutions or better bounds. In the following the potential of the solution methods of this thesis are evaluated in this context.

- TSD has shown to be a promising method for high school timetabling, and it has shown to be the best method for providing lower bounds for the DHSTP. A downside of the

method is that some constraints cannot be optimally handled, due to the structure of the problems and the current state-of-the-art theory of the TSD. Future research should clearly state the theory of TSD, such that it is completely separated from its application to the given optimization problem. Another topic for future research is a general algorithm for generating the necessary Hall inequalities, i.e. an algorithm which is not problem-specific. This would make the TSD more generally applicable.

Another topic for future research is further decomposition of the models yielded by the TSD approach. This could for instance be the application of a B&P approach to the Stage I model of both the DHSTP and GHSTP (as these models have shown to be hard to solve in practice). This may lead to improved solutions.

- The proposed matheuristic provides good solutions for both the GHSTP and the DHSTP. However, on the larger instances the overhead of invoking the MIP solver is high, which significantly harms performance. A way to handle this could be to not consider the entire MIP model at all times, but rather to either divide it into separate models or gradually add variables/constraints to the model.

- The formulation of the GMPP is capable of handling a wide range of problems, and it should be tested on other problems than the CTP. This could for instance be datasets of the DHSTP or the GHSTP. GMPP is also capable of providing bounds (which were of high quality in case of the CTP), which would be beneficial for both the DHSTP and the GHSTP.

Furthermore, it would be interesting to apply the developed matheuristic and extended TSD to other timetabling problems as well. This could for instance be university course-timetabling or examination timetabling.

Even though the developed models of the considered timetabling problems are believed to contain all constraints required in practice, future political decisions can change this. Changes in the problem-definition will possibly require changes to the solution methods as well. Therefore continuous research may be required in the future for these optimization problems.

Bibliography

R. K. Ahuja, Özlem Ergun, J. B. Orlin, and A. P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75 – 102, 2002. ISSN 0166-218X.

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, March 1998. ISSN 0030-364X.

R. E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 19–24 2012.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151, 2011. ISSN 1568-4946.

E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 457–474. Springer US, 2003. ISBN 978-1-4020-7263-5.

E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward. A classification of hyper-heuristic approaches. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 449–468. Springer US, 2010. ISBN 978-1-4419-1663-1.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

M. Caserta and S. Voss. Metaheuristics: Intelligent problem solving. In V. Maniezzo, T. Stützle, and S. Voss, editors, *Matheuristics*, volume 10 of *Annals of Information Systems*, pages 1–38. Springer US, 2010. ISBN 978-1-4419-1305-0.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2000.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449 – 467, 2001. ISSN 0377-2217.

J.-K. Hao and U. Benlic. Lower bounds for the itc-2007 curriculum-based course timetabling problem. *European Journal of Operational Research*, 212(3):464 – 472, 2011. ISSN 0377-2217.

INFORMS. What is operations research. `https://www.informs.org/About-INFORMS/What-is-Operations-Research` [Accessed 25/9-2013], Sep. 2013.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

J. H. Kingston. Educational timetabling. In A. S. Uyar, E. Ozcan, and N. Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013a. ISBN 978-3-642-39303-7.

J. H. Kingston. High school timetable file format specification: Constraints. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi?op=spec&part=constraints` [Accessed 12/11-2013], 2013b.

J. H. Kingston. High school timetable data format specification. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi?op=spec` [Accessed 12/11-2013], 2013c.

P. Laarhoven and E. Aarts. Simulated annealing. In *Simulated Annealing: Theory and Applications*, volume 37 of *Mathematics and Its Applications*, pages 7–15. Springer Netherlands, 1987. ISBN 978-90-481-8438-5.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

A. Lodi. The heuristic (dark) side of mip solvers. In E.-G. Talbi, editor, *Hybrid Metaheuristics*, volume 434 of *Studies in Computational Intelligence*, pages 273–284. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-30670-9.

V. Maniezzo, T. Stützle, and S. Voß. Matheuristics: Hybridizing metaheuristics and mathematical programming. *Annals of Information Systems*, 10, 2009a.

V. Maniezzo, S. Voss, and P. Hansen. Special issue on mathematical contributions to metaheuristics. *Journal of Heuristics*, 15(3), June 2009b.

B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

I. Osman and G. Laporte. Metaheuristics: A bibliography. *Annals of Operations Research*, 63 (5):511–623, 1996. ISSN 0254-5330.

B. Paechter, L. M. Gambardella, and O. Rossi-Doria. The first international timetabling competition.
`http://www.idsia.ch/Files/ttcomp2002/` [Accessed 25/9-2013], 2002.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

G. Post. International timetabling competition 2011.
`http://www.utwente.nl/ctit/hstt/itc2011/welcome/` [Accessed 25/9-2013], Sep. 2013a.

G. Post. Benchmarking project for (high) school timetabling.
`http://www.utwente.nl/ctit/hstt/` [Accessed 25/9-2013], Aug. 2013b.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

G. Post, L. Gaspero, J. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39 (3):728 – 735, 2012. ISSN 0305-0548.

J. Romrös and J. Homberger. An evolutionary algorithm for high school timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 485–488. SINTEF, 2012.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

D. Ryan. It is time to enjoy the best of both worlds. In *The 46th ORSNZ Conference*, Victoria University of Wellington, New Zealand, 10-11 December 2012.

M. Salazar-Aguilar, A. Langevin, and G. Laporte. An adaptive large neighborhood search heuristic for a snow plowing problem with synchronized routes. In J. Pahl, T. Reiners, and S. Voss, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 406–411. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-21526-1.

H. A. Taha. *Operations Research An Introduction (Sixth Edition)*. Prentice Hall, 1997.

The Ministry of Education in Denmark. Analyse om øget anvendelse af it på selvejende uddannelsesinstitutioner under undervisningsministeriet – forslag til forbedringer af studieadministrative opgaver og processer, June 2009. In Danish.

The Ministry of Education in Denmark. Four upper secondary education programmes in denmark.
`http://eng.uvm.dk/Education/Upper-Secondary-Education/Four-Upper-Secondary-Education-Programmes-in-Denmark` [Accessed 11/11-2013], 2013.

P. Wang, G. Reinelt, and Y. Tan. Self-adaptive large neighborhood search algorithm for parallel machine scheduling problems. *Journal of Systems Engineering and Electronics*, 23 (2):208–215, 2012.

Part II

Scientific Papers

Chapter 6 Paper A

Integer Programming for the Generalized (High) School Timetabling Problem

Simon Kristiansen[1,2], Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract Recently the XHSTT format for (High) School Timetabling was introduced, which provides a uniform way of modeling problem instances and corresponding solutions. The format supports a big variety of constraints, and currently 38 real-life instances from 11 different countries are available. Thereby the XHSTT format serves as a common ground for researchers within this area. This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a Mixed-Integer linear Programming (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results show that our approach is able to find previously unknown optimal solutions for 2 instances of XHSTT, and proves optimality of 4 known solutions. For the instances not solved to optimality, new non-trivial lower bounds were found in 11 cases, and new best-known solutions were found in 9 cases. Furthermore the approach is shown to be competitive with the finalist of Round 2 of the International Timetabling Competition 2011.

6.1 Introduction

The problem of scheduling lectures to time slots and/or resources at high schools is known as the *High School Timetabling* (HST) problem. This is an important problem for high schools in many countries, and a large amount of different solution approaches have been proposed, see the survey Schaerf (1999).

It is well recognized that the specifications of the HST problem varies significantly depending on the country of which the problem originates, and that the problem in general is hard to solve. With the introduction of the XHSTT format (Post et al., 2012a), a large number of instances from various origins became publicly available in standardized form. The format is based on the *Extensible Markup Language* (XML) standard, and all instances are available online (Post, 2013b). One purpose of the format is to serve as a common test-bed for *school timetabling*, in an attempt to promote research within this area. In this context, "school timetabling" denotes the area covering high school timetabling and *university course timetabling*, as the format has also been shown capable of modeling some instances of the latter problem (see Kingston (2013a) for an overview of educational timetabling problems).

This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a *Mixed-Integer linear Programming* (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results are performed for all the real-life instances currently available. Thereby we are able to find previously unknown optimal solutions, and prove optimality of already known solutions.

To the best of our knowledge, all previous solution methods for the XHSTT format have been heuristic in nature. Therefore no proof of optimality has been made for any instance, except for those instances where a solution with objective value 0 is known, since 0 is a trivial lower bound for any XHSTT instance. The obvious advantage of *Integer Programming* (IP) over heuristic methods is the capability to issue certificates of optimality. Therefore it is remarked that a big advance within general-purpose MIP solvers has happened in recent years, see e.g. Bixby (2012). Even though the MIP we will present is inevitable complex in nature, it will be shown that it can be used to find optimal solutions for several instances of the XHSTT archive `ALL_INSTANCES`. For those instances where an optimal solution cannot be found, we are able to show a non-trivial lower bound on optimum in the majority of cases. These are significant results for high school timetabling in general.

The outline of this paper is as follows. Section 6.2 presents related literature. Section 6.3 presents the MIP model of XHSTT. Section 6.4 describes computational results. Finally, Section 6.5 concludes and describes future research possibilities.

6.2 Related Literature

The Third International Timetabling Competition (ITC2011) considered the HST Problem, based on instances of the XHSTT format (Post et al., 2012b). Four teams were part of the final round: The overall winner (Team Goal) used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al., 2012). Participant from the University of Nottingham (HySTT) used a method based on Hyper-heuristics (Kheiri et al., 2012). Team Lectio used Adaptive Large Neighborhood Search (ALNS) (Sørensen et al., 2012). Romrös and Homberger (2012) (Team HFT) used an Evolutionary Algorithm. The results of the competition can be found at the official homepage of ITC2011 (Post (2013a)).

Pimmer and Raidl (2013) describe a 'timeslot-filling' heuristic for XHSTT, which iteratively fills selected timeslots with sets of events. Two state-of-the-art solutions were found for instances of the archive `XHSTT-2012`. Ter Braak (2012) presents a Hyper-heuristic and several other heuristics for the XHSTT.

Valouxis et al. (2012) describe a two-phase approach based on MIP used to solve the Greek case of the HST problem. This includes two instances which are part of the XHSTT project, and which were both solved to optimality (solutions were found with an objective value of 0).

In terms of Integer Programming and HST problems not based on XHSTT, the following contribution are mentioned: Santos et al. (2012) present a *Column Generation* approach for establishing bounds for a set of datasets originating from Brazil. Birbas et al. (2009) present an approach for Greek datasets where the *Shift Assignment Problem* is solved first, and the timetable is constructed on the basis on these work-shifts for teachers. The paper of Sørensen and Stidsen (2013) describes a complex MIP of the Danish case of high school timetabling, and establishes computational results for 100 real-life instances. Avella et al. (2007) present an algorithm based on Very Large-Scale Neighborhood search where the neighborhood is explored by a MIP, for Italian cases of high school timetabling.

6.3 Problem Description and a Mixed Integer Programming Formulation

In this section a brief description of the specifications of the XHSTT format is given, and a MIP model is formulated. The entire documentation of XHSTT is available at Post (2013b). We do not intend to describe all properties of the format, but only those necessary to formulate the MIP.

An instance of XHSTT consists of *times* (denoted $\mathcal{T}$ in the following), *time groups* (denoted $\mathcal{TG}$), *resources* (denoted $\mathcal{R}$), *events* (denoted $\mathcal{E}$), *event groups* (denoted $\mathcal{EG}$) and *constraints* (denoted $\mathcal{C}$). An event $e \in \mathcal{E}$ has a duration $D_e \in \mathbb{N}$, and a number of *event resources* which we each denote $er \in e$. An event resource defines the requirement of the assignment of a resource to the event, and this resource can be specified to be preassigned. If the resource is not preassigned, a resource of proper *type* must be assigned. Furthermore an event resource er can undertake a specific role_{er}, which is used to link the event resource to certain constraints.

It is the job of any solver for XHSTT to decide how each event should be split into *sub-events*. A sub-event se is defined as a fragment of a specific event $e \in \mathcal{E}$, has a duration $D_{se} \leq D_e$, and inherits the requirement of resources defined by the event, such that each sub-event has the exact same resource requirements as the event. Let $\mathcal{SE}$ denote the entire set of sub-events, and let $se \in e$ specify that sub-event se is part of event e. The total duration of all sub-events for event $e \in \mathcal{E}$ in a solution cannot exceed D_e. In our model formulation we create the 'full set' of sub-events with different lengths, i.e. all possible combinations of sub-events for a given event can be handled. E.g. 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. As a constraint it is then specified that the summed duration of the *active* sub-events in a solution must equals 4. A sub-event is active if it is assigned a starting time or a non-preassigned resource. An active sub-event is analogous to the concept of *solution events* defined in the XHSTT documentation.

The times $\mathcal{T}$ are ordered in chronological order, and we let $\rho(t)$ denote the index number of time t in $\mathcal{T}$. A time group $\mathcal{TG}$ defines a set of times, and we let $t \in tg$ denote that time t is part of time group tg.

Each constraint $c \in \mathcal{C}$ is of a specific type, and the set $\mathcal{C}$ can contain several constraints of the same type. Each constraint applies to certain events, event groups or resources, and penalizes certain characteristics of the timetable for these entities.

The following notation shorthand is made: By the notions $e \in c$, $r \in c$, $eg \in c$ we denote that constraint $c \in \mathcal{C}$ applies to event $e \in \mathcal{E}$, resource $r \in \mathcal{R}$, and event group $eg \in \mathcal{EG}$, respectively.

The set of resources and times are both extended with a dummy-index, denoted the *dummy-resource* r_D and the *dummy-time* t_D, respectively. These are necessary to ensure feasibility as we create all combinations of sub-events for each event, and not all of these can be assigned a time or the required resources without the duration of the active sub-events exceeding the duration of the event. Thereby these dummy-elements in fact represent that an event resource is *not* assigned a resource, and that a sub-event is *not* assigned a starting time, respectively.

6.3.1 Objective Function

Each XHSTT constraint penalizes timetables with certain characteristics, which contributes to the objective function of the MIP. Each constraint $c \in \mathcal{C}$ has a set of *point-of-applications* (indexed by $p \in c$). With each point-of-application is associated a set of *deviations* (indexed by $d \in p$), and each deviation has a non-negative cost associated with it. How this cost is calculated depends on the constraint type. The cost of a point-of-application is found on basis of the cost

Table 6.1: Different constraint types in the XHSTT format (Post et al., 2012b)

Constraint	Description
Assign Resource	Event resource should be assigned a resource
Assign Time	Event should be assigned a time
Split Events	Event should split into a constrained number of sub-events
Distribute Split Events	Event should split into sub-events of constrained durations
Prefer Resources	Event resource assignment should come from resource group
Prefer Times	Event time assignment should come from time group
Avoid Split Assignments	Set of event resources should be assigned the same resource
Spread Events	Set of events should be spread evenly through the cycle
Link Events	Set of events should be assigned the same time
Order Events	Set of events should be ordered
Avoid Clashes	Resource's timetable should not have clashes
Avoid Unavailable Times	Resource should not be busy at unavailable times
Limit Idle Times	Resource's timetable should not have idle times
Cluster Busy Times	Resource should be busy on a limited number of days
Limit Busy Times	Resource should be busy a limited number of times each day
Limit Workload	Resource's total workload should be limited

of the deviations, and is influenced by an indication on the constraint whether the constraint is a *hard* or a *soft* constraints, the *weight* of the constraint ($\omega_c \in \mathbb{N}$) and an indication of which *CostFunction* to use. For each constraint $c \in \mathcal{C}$ we let the variable $s_{c,p,d} \in \mathbb{N}$ be the penalty value of the deviation $d \in p$ of the point-of-application $p \in c$. The set of point-of-applications and deviations should be understood in an abstract context; E.g. depending on the type of the constraint, a point-of-application could be an event, a resource, etc., and likewise for the deviations.

The objective of a solution consists of a value for both the hard constraints (denoted *hard cost*) and a value for the soft constraints (denoted *soft cost*). Usually the objective value of a solution is written as (hard cost, soft cost). The hard cost always takes priority over the soft cost, i.e. solutions are first ranked on their hard cost, and secondly on the soft cost. How this type of objective function is handled in context of a MIP is described in Section 6.3.4.

The cost of a constraint $c \in \mathcal{C}$ which contains slack variable $s_{c,p,d}$ is denoted $f(s_{c,p,d})$,

$$f(s_{c,p,d}) = \omega_c \cdot \text{CostFunction}(s_{c,p,d}) \tag{6.1}$$

Five different types of CostFunction are allowed. The most trivial one is `Sum`, which simply sums the penalty value of all deviations for all point-of-applications. In the following each CostFunction is formulated in linear terms. Let the variable $\text{obj}_c \in \mathbb{N}_0$ denote the value of the of the CostFunction of constraint $c \in \mathcal{C}$.

- **Sum:** Sum the deviations.

$$\text{obj}_c = \sum_{p \in c, d \in p} s_{p,d,c} \quad \forall c \in \mathcal{C} \tag{6.2}$$

- **SumSquare:** Sum the squares of the deviations.
  To cope with this non-linear cost function, the variable $s_{c,p,d,i} \in \{0,1\}$ is introduced, which takes value 1 if the deviation $d \in p$ of the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has

the penalty $i \in \mathcal{I}$, and 0 otherwise. The objective value is defined as follows:

$$\text{obj}_c = \sum_{p \in c, d \in p, i \in \mathcal{I}} i^2 \cdot s_{c,p,d,i} \quad \forall c \in \mathcal{C} \tag{6.3}$$

However we also need to make sure that only a single integer value is selected,

$$\sum_{i \in \mathcal{I}} s_{c,p,d,i} = 1 \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{6.4}$$

The amount of elements in the set $\mathcal{I}$ determines the maximum possible penalty for a deviation, and thereby influence the maximum possible penalty for a constraint. To maintain optimality of the model, it is therefore important that the size of $\mathcal{I}$ is selected sufficiently large. This is elaborated in Section 6.4.

- **SquareSum:** Square the sum of deviations.
The binary slack variable $u^{\text{squaresum}}_{c,p,j} \in \{0,1\}$ is introduced, which takes value 1 if the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has the deviation $j \in \mathcal{J}$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, j \in \mathcal{J}} j^2 \cdot u^{\text{squaresum}}_{c,p,j} \quad \forall c \in \mathcal{C} \tag{6.5}$$

$$\sum_{j \in \mathcal{J}} u^{\text{squaresum}}_{c,p,j} = 1 \quad \forall c \in \mathcal{C}, p \in c \tag{6.6}$$

$$\sum_{d \in p} s_{p,d,c} = \sum_{j \in \mathcal{J}} j \cdot u^{\text{squaresum}}_{c,p,j} \quad \forall c \in \mathcal{C}, p \in c \tag{6.7}$$

Like the set $\mathcal{I}$, the size of the set $\mathcal{J}$ must be selected sufficiently large to maintain optimality, see Section 6.4.

- **SumStep:** This penalizes by the number of positive deviations, irrespective of their value.
The binary variable $u^{\text{sumstep}}_{c,p,d} \in \{0,1\}$ is introduced, which takes value 1 iff $s_{c,p,d} > 0$ for constraint $c \in \mathcal{C}$, point-of-application $p \in c$ and deviation $d \in p$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, d \in p} u^{\text{sumstep}}_{c,p,d} \quad \forall c \in \mathcal{C} \tag{6.8}$$

$$M \cdot u^{\text{sumstep}}_{c,p,d} \geq s_{c,p,d} \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{6.9}$$

where $M \in \mathbb{N}$ is some sufficiently large number.

- **StepSum:** This CostFunction penalizes by investigating whether the constrain contains at least one positive deviation. If this is not the case, the penalty is 0.
The binary variable $u^{\text{stepsum}}_{c} \in \{0,1\}$ is introduced, which takes value 1 if there exists at least one positive deviation for the constraint $c \in \mathcal{C}$, and 0 otherwise.

$$\text{obj}_c = u^{\text{stepsum}}_{c} \quad \forall c \in \mathcal{C} \tag{6.10}$$

$$M \cdot u^{\text{stepsum}}_{c} \geq s_{c,p,d} \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{6.11}$$

where $M \in \mathbb{N}$ is some sufficiently large number.

6.3.2 Mixed-Integer Programming Formulation

In this section the variables and the constraints of the MIP are described. As a basis for our approach is the variable $x_{se,t,er,r} \in \{0,1\}$, which takes value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time and resource $r \in er$ is assigned to event resource $er \in se$, and 0 otherwise. To simplify notation, and to reduce the amount of non-zeros in the MIP, three auxiliary variables are introduced which all 'inherits' their values directly from $x_{se,t,er,r}$. Let the binary variable $y_{se,t} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time, and 0 otherwise. The variable $v_{t,r} \in \mathbb{N}_0$ denotes the number of times resource r is used in time t by any set of sub-events. Let variable $w_{se,er,r} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is assigned resource $r \in \mathcal{R}$ for event resource er, and 0 otherwise.

6.3.2.1 Base Constaints

Besides all the constraints described in the specifications of the XHSTT, some basic constraints are needed to ensure feasibility. First of all we need to make sure that a sub-event is assigned only one starting time and that the number of resource assigned is exactly the same as the number of event resources of the event.

$$\sum_{t \in \mathcal{T}, r \in er} x_{se,t,er,r} = 1 \quad \forall se \in \mathcal{SE}, er \in se \tag{6.12}$$

The following constrains variable $y_{se,t}$, and together with (6.12) ensures that a sub-event is not spread across multiple times. We denote by $|er|_{se}$ the amount of event resources of sub-event $se \in \mathcal{SE}$.

$$\sum_{er \in se, r \in er} x_{se,t,er,r} = |er|_{se} \cdot y_{se,t} \quad \forall se \in \mathcal{SE}, t \in \mathcal{T} \tag{6.13}$$

The link to variable $v_{t,r}$ is shown in eq. (6.15). For time $t \in \mathcal{T}$ and $se \in \mathcal{SE}$ is found the set of possible starting-times for se which will cause resource $r \in \mathcal{R}$ to be used in time $t \in \mathcal{T}$. Let the set $T^{\text{start}}_{se,t} \subseteq \mathcal{T}$ be the set of times which sub-event se lies in if it is assigned starting time t, i.e.

$$T^{\text{start}}_{se,t} = \{t' \in \mathcal{T} \setminus t_D \mid \rho(t) - D_{se} + 1 \leq \rho(t') \leq \rho(t)\} \tag{6.14}$$

$$\sum_{se \in \mathcal{SE}, er \in se, t' \in T^{\text{start}}_{se,t}} x_{se,t',er,r} = v_{t,r} \quad \forall t \in \mathcal{T} \setminus t_D, r \in \mathcal{R} \tag{6.15}$$

The link to variable $w_{se,er,r}$ looks as follows:

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \quad \forall se \in \mathcal{SE}, er \in se, r \in er \tag{6.16}$$

A sub-event cannot be assigned a start time if there is not enough continuous times after the start time to fulfill the duration, ensured by the constraint:

$$y_{se,t} = 0 \quad \forall se \in \mathcal{SE}, t \in \mathcal{T} \setminus t_D, \rho(t) + D_{se} - 1 > |\mathcal{T}| \tag{6.17}$$

Active Sub-events

As we create all possible sub-events for a given event, only a subset of these should be active in the final solution. The binary variable $u_{se} \in \{0,1\}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is active and 0 otherwise. Recall that a sub-event is active if its assigned a starting time, or if is assigned at least one non-preassigned resource. Let the parameter $PA_{er} \in \{0,1\}$ take value 1 if event resource er has a preassigned resource, and 0 otherwise. The following constraints are imposed.

$$\sum_{r \in er \setminus r_D} w_{se,er,r} \leq u_{se} \quad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{6.18}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} \leq u_{se} \quad \forall se \in \mathcal{SE} \tag{6.19}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} + \sum_{r \in er \setminus r_D} w_{se,er,r} \geq u_{se} \quad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{6.20}$$

Constraint (6.20) is necessary to ensure events are not set as active, even though they do not meet the required criteria.

The duration of active sub-events for a given event must be exactly the same as the total duration of the event (by definition of a valid XHSTT solution),

$$\sum_{se \in e} D_{se} \cdot u_{se} = D_e \quad \forall e \in \mathcal{E} \tag{6.21}$$

A number of constraints require that the value of a deviation $V \in \mathbb{N}$ should be within an upper-limit $\overline{B}_c \in \mathbb{N}$ and a lower-limit $\underline{B}_c \in \mathbb{N}$. This means that the penalty is defined as the amount which the value of a deviation exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$. To simplify notation for these cases, we introduce the function $\mathcal{U}_{\underline{B}_c,\overline{B}_c} V$, which is defined as follows:

$$s \geq \mathcal{U}_{\underline{B}_c,\overline{B}_c} V \Rightarrow \begin{cases} s \geq V - \overline{B}_c \\ s \geq \underline{B}_c - V \end{cases} \tag{6.22}$$

Thereby the slack-variable s is forced to take the actual value of the imposed penalty.

A resource is *busy* at some time if it attends at least one solution event at that time, and busy at some time group if it is busy at one or more times within times of that time group. Let variable $q_{r,t} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time $t \in \mathcal{T}$, and 0 otherwise. Similarly, let the binary variable $p_{r,tg} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time group $tg \in \mathcal{TG}$, and 0 otherwise. The values of the two variables are determined by the following constraints.

$$|\mathcal{SE}| \cdot q_{r,t} \geq v_{t,r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{6.23}$$

$$q_{r,t} \leq v_{t,r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{6.24}$$

$$p_{r,tg} \geq q_{r,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{6.25}$$

$$p_{r,tg} \leq \sum_{t \in tg} q_{r,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{6.26}$$

Constraints (6.23) and (6.25) establishes lower bounds for the variables $q_{r,t}$ and $p_{r,tg}$, i.e. ensures that these must take value 1 in case the resource is actually busy in the respective time/time

group. Constraints (6.24) and (6.26) are necessary to ensure that in case the resource is in fact *not* busy in the respective time/time group, variables $q_{r,t}$ and $p_{r,tg}$ must take value 0.

In the following the constraint types of the XHSTT documentation are formulated one by one. Each constraint type is described in brief terms, and we refer to Kingston (2013c) for more details. The formulation of these constraints in terms of a Mixed-Integer Linear Programming model has not been published before. We let the 'pseudo-set' $\bar{\mathcal{C}} \subseteq \mathcal{C}$ denote constraints of a certain type depending on the context, for instance the set of all *assign resource* constraints. Furthermore we in the following make use of the general slack variable $s_{c,p,d}$, and will for each type of constraint implicitly define a corresponding slack variable with the appropriate indices for point-of-applications and deviations.

6.3.2.2 Assign Resources

Applies to: Events
Point-of-application: Event-resource
An *Assign Resource* constraint penalizes event resources that are not assigned resources. Specifically, the deviation at one point of application (an event resource with the appropriate role) is the sum of the duration of the sub-events of the respective event which are not assigned a resource. The cost of this constraint is given by:

$$D_e - \sum_{\substack{se \in e \\ r \in er \setminus r_D}} D_{se} \cdot w_{se,er,r} = s^{\text{assignres}}_{c,er} \quad \forall c \in \bar{\mathcal{C}}, e \in c, er \in e, \text{role}_{er} = \text{role}_c \tag{6.27}$$

6.3.2.3 Assign Time

Applies to: Events
Point-of-application: Events
The assign time constraint penalizes sub-events which are not assigned times. The deviation at one point of application is the total duration of those sub-events derived from a specific event that are not assigned a time.

$$D_e - \sum_{\substack{t \in \mathcal{T} \setminus t_D \\ se \in e}} D_{se} \cdot y_{se,t} = s^{\text{assigntime}}_{c,e} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{6.28}$$

6.3.2.4 Split Events

Applies to: Events
Point-of-application: Events
A *Split Event* constraint places limits on the number of sub-events that may be derived from a given event, and on their duration. Let the parameters $\underline{B}^{\text{amount}}_c \in \mathbb{N}$ and $\overline{B}^{\text{amount}}_c \in \mathbb{N}$ denote the minimum and maximum amount of sub-events which is used for a given event, respectively. And let $\underline{B}^{\text{dur}}_c \in \mathbb{N}$ and $\overline{B}^{\text{dur}}_c \in \mathbb{N}$ be the minimum and maximum duration a sub-event can have for a given event, respectively.

The cost of this constraint is given by the number of sub-events whose duration is less than $\underline{B}^{\text{dur}}_c$ or greater than $\overline{B}^{\text{dur}}_c$, and the amount by which the number of sub-events fall short of $\underline{B}^{\text{amount}}_c$ or exceed $\overline{B}^{\text{amount}}_c$. The following constraints are imposed:

$$\mathcal{U}_{\underline{B}^{\text{amount}}_c, \overline{B}^{\text{amount}}_c} \sum_{se \in e} u_{se} \leq s^{\text{spliteventamount}}_{c,e} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{6.29}$$

$$\sum_{\substack{se \in e \\ \underline{B}_c^{\text{dur}} > D_{se} \vee \overline{B}_c^{\text{dur}} < D_{se}}} u_{se} = s_{c,e}^{\text{spliteventdur}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{6.30}$$

The full deviation for constraint $c \in \bar{\mathcal{C}}$ and event $e \in c$ is given by $s_{c,e}^{\text{spliteventdur}} + s_{c,e}^{\text{spliteventamount}}$.

6.3.2.5 Distribute Split Event

Applies to: Events
Point-of-application: Events
The *Distribute Split Event* constraints set limits on the number of sub-events which 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 which may be derived from a given event.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{\substack{se \in e \\ D_{se} = D_c}} u_{se} \leq s_{c,e,er}^{\text{distsplitevent}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{6.31}$$

6.3.2.6 Prefer Resources

Applies to: Events
Point-of-application: Event-resources
This constraint defines that an event resource has different preferences for certain resources. The deviation is calculated by taking all the solution resources derived from the event resource that are assigned a resource that is not one of the preferred resources, and summing the duration of the sub-events that those resources lie in. Let $r \in c$ denote a preferred resources.

$$\sum_{\substack{se \in e \\ r \notin c, r \neq r_D}} D_{se} \cdot w_{se,er,r} = s_{c,er}^{\text{preferres}} \quad \forall c \in \bar{\mathcal{C}}, e \in c, er \in e, PA_{er} = 0, \text{role}_{er} = \text{role}_c \tag{6.32}$$

6.3.2.7 Prefer Times constraints

Applies to: Events
Point-of-application: Events
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 is assigned a time which is not one of the preferred time. 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 $t \in c$ denote a preferred time.

$$\sum_{\substack{se \in e \\ t \notin c, t \neq t_D \\ D_c = D_{se}}} D_{se} \cdot y_{se,t} = s_{c,e}^{\text{prefertime}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{6.33}$$

6.3.2.8 Avoid Split Assignments

Applies to: Evengroups
Point-of-application: Eventgroups
Each solution resource can only have one resource assigned. However, when an event is split into sub-events, each of its event resources is split into several solution resources, and a different

resource may be assigned to each of these solution resources. This constraint penalizes the assignment of different resource to these solution resources. The constraint examines all the solution resources derived from those event resources, and calculates the number of distinct resources assigned to them, ignoring unassigned solution resources. The deviation is the amount by which this number exceeds 1. Let variable $k_{c,eg,r} \in \{0,1\}$ take value 1 if event e is assigned to resource r with respect to avoid split assignment constraint c, and 0 otherwise.

$$\sum_{\substack{er \in e, PA_{er}=0 \\ \text{role}_c = \text{role}_{er}}} w_{se,er,r} \leq k_{c,eg,r} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, e \in eg, se \in e \tag{6.34}$$

$$\sum_{r \in \mathcal{R}} k_{c,eg,r} - 1 \leq s^{\text{avoidsplitass}}_{c,eg} \quad \forall c \in \bar{\mathcal{C}}, eg \in c \tag{6.35}$$

6.3.2.9 Spread Events

Applies to: Eventgroups
Point-of-application: Eventgroups
The *Spread Event* constraint has a deviation for each time group $tg \in c \in \bar{\mathcal{C}}$. Let $\underline{B}_{c,tg}$ and $\overline{B}_{c,tg}$ be the minimum and maximum number of sub-events of a given event which can be placed in time group tg of constraint c, respectively. The deviation for each time group is given by the amount of which the number of sub-events for the given event which fall short of $\underline{B}_{c,tg}$ or exceeds $\underline{B}_{c,tg}$.

$$\mathcal{U}_{\underline{B}_{c,tg}, \overline{B}_{c,tg}} \sum_{\substack{se \in e \in eg \\ t \in tg}} y_{se,t} \leq s^{\text{spreadevent}}_{c,eg,tg} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, tg \in c \tag{6.36}$$

6.3.2.10 Link Events

Applies to: Eventgroups
Point-of-application: Eventgroups
A *Link Event* constraint specifies that some events should be assigned the same times. For each event of a given event group we build the set of times that the sub-events derived from that event are running (not just starting times). The deviation is then the number of times that appear in at least one of these sets but not in all of them. Let variable $o_{e,t} \in \{0,1\}$ take value 1 if at least one sub-event of event $e \in c \in \bar{\mathcal{C}}$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Let variable $l_{eg,t} \in \{0,1\}$ take value 1 if at least one event of event group $eg \in c$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Constraints (6.37) and (6.39) ensure that these variables take correct values. The slack of Link Events constraints is defined in (6.40). Constraint (6.38) is necessary to restrict $o_{e,t}$ to take value 1 in cases where the event is in fact not assigned to the particular time, which would avoid the penalty given by constraint (6.40), if any.

$$\sum_{t' \in T^{\text{start}}_{se,t}} y_{se,t'} \leq o_{e,t} \quad \forall e \in \mathcal{E}, se \in e, t \in \mathcal{T} \setminus t_D \tag{6.37}$$

$$\sum_{\substack{se \in e \\ t' \in T^{\text{start}}_{se,t}}} y_{se,t'} \geq o_{e,t} \quad \forall e \in \mathcal{E}, t \in \mathcal{T} \setminus t_D \tag{6.38}$$

$$l_{eg,t} \geq o_{e,t} \quad \forall eg \in \mathcal{EG}, e \in eg, t \in \mathcal{T} \setminus t_D \tag{6.39}$$

$$l_{eg,t} - o_{e,t} \leq s^{\text{linkevent}}_{c,eg,t} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, e \in eg, t \in \mathcal{T} \setminus t_D \tag{6.40}$$

6.3.2.11 Order Events

Applies to: Pair of events
Point-of-application: Pair of events
An *Order Event* constraint specifies that the times two events are assigned should be in order, such that the first event ends before the second event starts. Let the parameters $\underline{B}_c \in \mathbb{N}$ and $\overline{B}_c \in \mathbb{N}$ be the minimum and maximum number of times that may separate the two events, respectively. Let $(e, e') \in c$ denote an *EventPair* which this constraint applies to. Let the variable $h^{\text{last}}_e \in \mathbb{N}$ be the ordinal number of the latest time assigned to any sub-event of event e. Let the variable $h^{\text{first}}_e \in \mathbb{N}$ be the ordinal number of the first assigned to any sub-event of event e'. The deviation is then given by the amount by which the difference between these two numbers exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$.

$$\rho(t) \cdot y_{se,t} + D_{se} \leq h^{\text{last}}_e \quad \forall c \in \bar{\mathcal{C}}, e \in c, se \in e, t \in \mathcal{T} \setminus t_D \tag{6.41}$$

$$|\mathcal{T}| - (|\mathcal{T}| - \rho(t)) \cdot y_{se,t} \leq h^{\text{first}}_e \quad \forall c \in \bar{\mathcal{C}}, e \in c, se \in e, t \in \mathcal{T} \setminus t_D \tag{6.42}$$

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c}(h^{\text{last}}_{e'} - h^{\text{first}}_e) \leq s^{\text{orderevents}}_{c,(e,e')} \quad \forall c \in \bar{\mathcal{C}}, (e, e') \in c \tag{6.43}$$

6.3.2.12 Avoid Clashes

Applies to: Resources
Point-of-application: Resources
These constraints specify that certain resources should have no clashes, i.e. they should not be assigned two or more events simultaneously. The constraint produces a set of deviations at each point of application (each resource). For each time a resource is assigned two or more solution resources, there is one deviation with a value equal to the number of solution resources minus one.

$$v_{t,r} - 1 \leq s^{\text{avoidclashes}}_{c,r,t} \quad \forall c \in \bar{\mathcal{C}}, r \in c, t \in \mathcal{T} \setminus t_D \tag{6.44}$$

6.3.2.13 Avoid Unavailable Times

Applies to: Resources
Point-of-application: Resource
An *Avoid Unavailable Times* constraint specifies that certain resources are unavailable for all events at certain times. The deviation is the number of unavailable times during which the resource attends at least one solution event. $t \in c$ denotes that t is an unavailable time for constraint $c \in \bar{\mathcal{C}}$.

$$\sum_{t \in c} q_{r,t} = s^{\text{unavailabletimes}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{6.45}$$

6.3.2.14 Limit Idle Times

Applies to: Resources
Point-of-application: Resources
A resource is idle at some time $t \in tg$ wrt. time group tg if it is not attending any sub-events at that time, but it is busy at some earlier time and at some later time in time group tg. The *Limit Idle Times* places limits on the number of idle times a resources may have. Let the variables $h^{\text{first}}_{r,tg} \in \mathbb{N}$ and $h^{\text{last}}_{r,tg} \in \mathbb{N}$ indicate the ordinal number of the first and the last time, respectively, where resource $r \in \mathcal{R}$ is busy in time group tg. Let $|tg|$ denote the amount of times in time group tg. Let the variable $h_{r,tg} \in \mathbb{N}$ denote the number of idle times of resource $r \in \mathcal{R}$ in time group $tg \in \mathcal{TG}$.

$$|tg| - (|tg| - \rho(t)) \cdot q_{r,t} \geq h^{\text{first}}_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{6.46}$$

$$\rho(t) \cdot q_{r,t} \leq h^{\text{last}}_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{6.47}$$

$$h^{\text{last}}_{r,tg} - h^{\text{first}}_{r,tg} + 1 - \sum_{t \in tg} q_{r,t} = h_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{6.48}$$

For each resource of the constraint the deviation is calculated as follows. Calculate the total amount of idle times for all times $tg \in c$, and find the amount which this summed value falls short of minimum $\underline{B}_c \in \mathbb{N}$ or exceeds maximum $\overline{B}_c \in \mathbb{N}$. The deviation is then given by the sum of these amounts.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} h_{r,tg} \leq s^{\text{idletimes}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{6.49}$$

6.3.2.15 Cluster Busy Times

Applies to: Resources
Point-of-application: Resources
A *Cluster Busy Times* constraint limits the number of time groups during which a resource may be busy. The deviation is given by the amount of by which the number of given time groups during which the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum, $\overline{B}_c \in \mathbb{N}$. Let $tg \in c$ denote a time group which this constraint applies to.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} p_{r,tg} \leq s^{\text{clusterbusy}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{6.50}$$

6.3.2.16 Limit Busy Times

Applies to: Resources
Point-of-application: Resources
The *Limit Busy Times* constraints places limits on the number of times a resource may be busy within some time groups. These constraints produces a set of deviation at each point-of-application, one for each given time group. The deviations are given by the amount by which the number of times of the given time group that the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum $\overline{B}_c \in \mathbb{N}$.

$$-|tg| \cdot (1 - p_{r,tg}) + \mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{t \in tg} q_{r,t} \leq s^{\text{limit busy}}_{c,r,tg} \quad \forall c \in \bar{\mathcal{C}}, r \in c, tg \in c \tag{6.51}$$

6.3.2.17 Limit Workload

Applies to: Resources
Point-of-application: Resources
A workload of a solution resource is given by $W_{e,se,er} = \frac{D_{se} \cdot L_{er}}{D_e}$, where $L_{er} \in \mathbb{N}$ is the workload of event resource er. The value is a floating-point number. A *Limit Workload Constraint* places limits on the total workload of solutions resources that certain resources are assigned to. The deviation of this constraint is the amount by which the total workload of the solution resources assigned to that resource falls short of $\underline{B}_c \in \mathbb{N}$ or exceeds $\overline{B}_c \in \mathbb{N}$, rounded up to the nearest integer.

$$\mathcal{U}_{\underline{B}_c,\overline{B}_c} \sum_{\substack{e \in c, t \in \mathcal{T} \setminus t_D \\ se \in e, er \in e}} W_{e,se,er} \cdot x_{se,t,er,r} \leq s^{\text{limitworkload}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{6.52}$$

6.3.3 Mixed-Integer Programming Model

Given the definitions of all constraint types of XHSTT, and their respective slack variables, the objective of the model can be stated as eq. (6.53), setting aside the fact that some constraints are hard-constraints and some are soft-constraints.

$$\begin{aligned} z = & f(s^{\text{assignres}}_{c,er}) + f(s^{\text{assigntime}}_{c,e}) \\ & + f(s^{\text{spliteventamount}}_{c,e} + s^{\text{spliteventdur}}_{c,e}) \\ & + f(s^{\text{distsplitevent}}_{c,e,er}) + f(s^{\text{preferres}}_{c,er}) + f(s^{\text{prefertime}}_{c,e}) \\ & + f(s^{\text{avoidsplitass}}_{c,eg}) + f(s^{\text{spreadevent}}_{c,eg,tg}) + f(s^{\text{linkevent}}_{c,eg,t}) \\ & + f(s^{\text{orderevents}}_{c,(e,e')}) + f(s^{\text{avoidclashes}}_{c,r,t}) \\ & + f(s^{\text{unavailabletimes}}_{c,r}) + f(s^{\text{idletimes}}_{c,r}) + f(s^{\text{clusterbusy}}_{c,r}) \\ & + f(s^{\text{limitbusy}}_{c,r,tg}) + f(s^{\text{limitworkload}}_{c,r}) \end{aligned} \tag{6.53}$$

The full MIP would therefore consists of minimizing z, subject to eqs. (6.12) to (6.52). However, we take a different approach, as described in the next section.

6.3.4 Solution Approach

Even though it would be natural to simply input the MIP to a generic solver, a different approach is taken, which takes advantage of the XHSTT objective function. In this approach, the model is solved in two steps, denoted Step 1 and Step 2 in the following.

By the definition of the XHSTT objective, hard constraints always take priority over soft constraints. Therefore the following approach is taken for solving the model: In Step 1, a MIP is build which only contains the hard constraints. This MIP is given as input to the MIP solver, which is ran until the given time limit is reached, or until the model is solved to optimality. The found objective value is the hard cost of the solution. In case the time limit is reached, all variables are fixed to their final value (i.e. the value they take in the best found solution), and all the soft constraints are added to identify the true cost of the found solution. In case the MIP is solved to optimality, Step 2 is performed: All soft constraints are added and the solution process is warm-started from its previous state, with the time limit set to what remains of the original time limit. Furthermore a constraint is added which ensures that the optimal value of the hard cost is kept. Let z^{hard} denote the sum of all slack variables belonging to the hard constraints.

The following constraint is added:

$$z^{\text{hard}} = \text{hard cost} \tag{6.54}$$

Now this extended MIP model is solved. The cost of the obtained solution, minus the hard cost found in Step 1, is the value of the soft cost. Notice that the nature of this solution method resembles *lexicographic multi-objective optimization*.

This approach takes advantage of the capability of MIPs to issue certificates of optimality. By this we mean that focus is put on the hard constraints until a solution is found with the optimal hard cost, and then we switch focus and consider the entire problem instance. If a heuristic solution method was used the inevitable question would be: When has sufficient effort been put into minimizing the hard cost?

6.4 Computational Results

This section presents computational results of the developed exact method, and has two primary intentions:

- How does the MIP compete with the heuristics of the ITC2011 round 2? Thereby the potential of this MIP approach can be evaluated on fair terms with well-performing heuristics. This is the subject of Section 6.4.1.
- Are we able to improve the best-known solutions for some instances, or even solve them to optimality? See Section 6.4.2.

All tests were run on a machine with an Intel i7 CPU clocked at 2.80 GHz and 12GB of RAM, running Windows 8 64 bit. In all cases the commercial state-of-art MIP solver Gurobi 5.5.0 was used. Two distinct sets of XHSTT instances have been used, both obtained from the XHSTT website (Post, 2013b). All obtained solutions have been verified as being valid using the evaluator *HSEval* (Kingston, 2013b).

As described in Section 6.3.1, an XHSTT objective consists of both a hard cost, and a soft cost, usually denoted (hard cost, soft cost). In case a solution has a hard cost of value 0, the objective is simply written as the soft cost, as is usually done in context of the XHSTT format.

As discussed in Section 6.3.1, the size of the sets $\mathcal{I}$ and $\mathcal{J}$ must be selected sufficiently high. Notice further that if the size of these sets is selected high, it can have a big impact on the amount of variables in the model. It would be possible to select these sizes based on the properties of the constraints having CostFunction `SumSquare` or `SquareSum`, however this is a quite complex operation as it must be derived based on each constraint-type. Instead we have selected $|\mathcal{I}| = 10$ $|\mathcal{J}| = 10$, such that the maximum possible penalty is $9^2 = 81$. This means that we cannot claim optimality for solutions with objective > 81. An easy fix for this issue is to simply perform a re-run of Gurobi if a solution is claimed as optimal, with the size of the sets set to a higher value. The same is applicable for lower bounds of value > 81. We consider this is an implementation detail, and it will be seen that in practice it has no impact of the obtained results.

6.4.1 International Timetabling Competition 2011

This section compares the exact method with the results obtained by the finalists in Round 2 of ITC2011. In this round the solver for each participating team was tested on 18 previously unknown instances from the archive `XHSTT-ITC2011-hidden`. The time limit for all instances was nominated to 1000 seconds, but the organizers provided a tool to benchmark machines

to find the machine-dependent equivalent of this time limit. On our machine this amended to 772 seconds. The possibility to benchmark machines facilitates a fair comparison with the competitors of ITC2011, except for the fact that the rules of ITC2011 did not allow the use of commercial software, which conflicts with our use of Gurobi. The aim of this section is therefore to demonstrate the potential of MIP in the context of timetabling (which is often overlooked), and not to claim how our approach would have positioned itself in ITC2011.

In terms of solver parameters, default settings are used, except for the *pseudo-parameter* `MIPFocus` which is set to value 1, emphasizing that we are mainly interested in finding incumbent solutions. Gurobi was only allowed to use a single CPU thread, as specified in the rules of ITC2011.

The participants of ITC2011 round 2 ran their algorithm 10 times on each instance, to eliminate the stochastic impact on the results. Since we are interested in the average performance of each participant for comparison, the following processing of the results was performed: For each instance and each participant, calculate both the average hard cost and the average soft cost, and round both to nearest integer. These numbers then denote how this participant performed on this instance.

Table 6.2 shows the obtained results. The value of "Avg. Ranking" was calculated as follows. Each solution method was ranked 1 to 5 on each instance, 1 being the best, and the average of these ranks was taken. According to this measure, the exact method of this paper is competitive with the methods used at ITC2011. Notice in particular that the exact method performs well on the smaller instances, and is generally not as competitive on the larger instances. On three instances the exact method gave the best results.

6.4.2 Aiming at Optimality

In attempt to produce new (optimal) solutions, the XHSTT archive `ALL_INSTANCES` was used, which contains 38 non-artificial instances. According to the website, this archive "contains all latest versions of the contributed instances". For 10 of the instances, a solution with cost 0 is already known, which constitutes an optimal solution by the definition of XHSTT. Hence these instances are skipped in this test. Notice that `ALL_INSTANCES` contains instances which originally came from `XHSTT-ITC2011-hidden`, but due to bug-fixes in some of the instances, we consider them as two separate sets of instances (by bug-fixes we mean altering of certain constraints, such that objective values are incomparable). We refer to (Post, 2013b) for instance-statistics.

This test was performed with the following setup: Gurobi is allowed to use all CPU cores (which is 8 in our case), and the time-limit is set to 24 hours for each instance. As initial solution for each instance, the current best known solution is provided. Default parameter settings of Gurobi were used. Table 6.3 shows the obtained results. A gap between an incumbent solution x and a lower bound LB is calculated by $\frac{|x-\mathrm{LB}|}{x}$.

For each instance a solution with XHSTT objective (H, S) is found, as well as a lower bound $(\underline{H}, \underline{S})$. By the definition of our solution method, we only have a lower bound on the soft cost $\underline{S}$ iff an optimal solution for the hard cost is known, i.e. $H = \underline{H}$. If a lower bound or an objective value is not found we write "-". Notice that even though we give the current best known solution as starting solution, Gurobi might still not find a solution for Step 1, usually in case the instance in question is of huge size. In Table 6.3, both the gap for the hard cost and the soft cost is shown (in case the required costs and lower bounds are available). Table 6.3 shows that our method obtains better solutions for 8 instances. 4 instances was solved to optimality, proving optimality of 3 previously known solutions and finding 1 new optimal solution. Furthermore, 11 new non-trivial lower bounds and 7 new best solutions have been established for the instances which were not solved to optimality.

Table 6.2: Performance of the MIP using same running time as specified in ITC2011. For each instance is listed the average solution found from each of the competitors of ITC2011, and the solution obtained by the MIP formulations. The best solutions are marked in **bold**. Objectives marked with * are optimal solutions.

Instance	GOAL	HySST	Lectio	HFT	Exact method
BrazilInstance2	(1, 62)	(1, 77)	**38**	(6, 190)	46
BrazilInstance3	124	118	152	(30, 283)	**39**
BrazilInstance4	(17, 98)	(4, 231)	**(2, 199)**	(67, 237)	(5, 286)
BrazilInstance6	(4, 227)	(3, 269)	**230**	(23, 390)	682
ElementarySchool	4	(1, 4)	**3**	(30, 73)	**3**
SecondarySchool2	**1**	23	34	(31, 1628)	(1604, 3878)
Aigio	**13**	(2, 470)	1062	(50, 3165)	(1074, 3573)
Italy_Instance4	**454**	6926	651	(263, 6379)	17842
KosovaInstance1	**(59, 9864)**	(1103, 14890)	(275, 7141)	(989, 39670)	(3626, 2620)
Kottenpark2003	**90928**	(1, 56462)	(50, 69773)	(209, 84115)	(8491, 6920)
Kottenpark2005A	**(31, 32108)**	(32, 30445)	(350, 91566)	(403, 46373)	(2567, 53)
Kottenpark2008	**(13, 33111)**	(141, 89350)	(209, 98663)	-	(14727, 5492)
Kottenpark2009	**(28, 12032)**	(38, 93269)	(128, 93634)	(345, 99999)	(17512, 140)
Woodlands2009	(2, 14)	(2, 70)	**(1, 107)**	(62, 338)	(1801, 705)
Spanish school	**894**	1668	2720	(65, 13653)	(1454, 11020)
WesternGreece3	**6**	11	(30, 2)	(15, 190)	25
WesternGreece4	**7**	21	(36, 95)	(237, 281)	81
WesternGreece5	**0**	4	(4, 19)	(11, 158)	15
Avg. Ranking	1.72	2.67	2.50	4.44	3.61

6.4.2.1 Alternative Formulation

The Limit Idle Times constraint is known to be difficult for solvers to handle (Dorneles et al. (2012)). In our formulation, this constraint is formulated using Big-M notation (constraints (6.46) and (6.47)), which can provide bad LP-relaxation, which in turn might slow down the solution process. Furthermore this constraint is part of most instances (29 of 38 instances in the `ALL_INSTANCES` archive), so an alternate formulation is proposed. The alternate formulation uses variable $h_{r,tg,t} \in \{0,1\}$ which takes value 1 if resource $r \in \mathcal{R}$ has an idle time in time $t \in tg$ in time group tg, and 0 otherwise. Constraints (6.46), (6.47) and (6.48) are replaced by

$$q_{r,t'} - q_{r,t} + q_{r,t''} - 1 \leq h_{r,tg,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t, t', t'' \in tg, \rho(t') < \rho(t) < \rho(t'') \tag{6.55}$$

This yields more rows in the MIP; for each time group $tg \in \mathcal{TG}$ the amount of additional constraints is $\binom{|tg|}{3}$. Furthermore, there is a great possibility that the amount of variables increases due to the extra dimension on the h variable. However, no Big-M notation is used.

Due to the possible big increase in the size of the model, this alternative formulation is only tested on the smaller instances from archive `ALL_INSTANCES`, skipping those instances in which the optimal solution was found in the previous test (Table 6.3). Since the goal is to achieve is good solutions as possible, we restart the procedure from the best found solution of Table 6.3 and run it for additional 24 hours. This test-setup means that we cannot compare the performance of the two formulations. Table 6.4 shows the obtained results. The table shows that this formulation is capable of finding 2 new optimal solutions. For the instances not solved to optimality, 6 lower bounds were improved, and new best solutions were found for 6 instances.

6.5 Conclusion

This paper has shown the first exact method for (High) School Timetabling instances in the XHSTT format. A solution method which takes advantage of the structure of the objective function of XHSTT has been proposed. For the most recent version of the archive `ALL_INSTANCES`, we were able to produce 2 new optimal solutions and prove optimality of 4 previously known solutions. For 11 other instances, new non-trivial lower bounds were shown. For the instances not solved to optimality, we were able to improve the best known solution in 9 cases.

Establishing optimal solutions and lower bounds is indeed a step forward for research within high school timetabling, and for the XHSTT format in particular. This gives researchers a possibility to compare their obtained solutions with an (optimal) lower bound, which is valuable for evaluating the quality of solutions.

As subjects for future research the following are mentioned. The MIP could be used in context of *Two-Stage Decomposition* (TSD), by first assigning times to events, and secondly assigning resources to event resources. Thereby the resource-assignments are done subject to the times assigned to events. Such an approach was used with great success in the paper of Lach and Lübbecke (2012) for the *Curriculum-based University Timetabling Problem* (the optimization problem used in the International Timetabling Competition 2007), and by Sørensen and Dahms (2014) for the real-world case of High School Timetabling in Denmark. In both of these papers, the TSD is theoretically capable of producing near-optimal results, even though the problem is split into two separate MIPs. However, the XHSTT case is possibly less suited for this type of decomposition as instances might contain a majority of constraints related to resource assignments. Since the assignments to times for events are performed in the first stage of the decomposition, and because these assignments cannot be altered when the resource-assignments are performed, a TSD approach would possibly be heuristic in nature. Obviously, if an XHSTT instance have all resources preassigned to event resources, a TSD would be unnecessary.

Our MIP formulation is exponential in size by the amount of sub-events in the instance, as all possible combinations of sub-events are generated. A better formulation would be less dependent on this amount. One could for instance solve the model iteratively, and 'inject' new sub-events in the model on-the-fly. Another possibility would be to consider a formulation which simulate sub-events by an integer variable which define the lengths of each respective active sub-event. Such improved formulations are subject for future research.

Bibliography

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

R. E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 19–24 2012.

M. ter Braak. A hyperheuristic for generating timetables in the xhstt format. Master's thesis, University of Twente, June 2012.

Á. P. Dorneles, O. C. de Araújo, S. Maria-Brazil, and L. S. Buriol. The impact of compactness requirements on the resolution of high school timetabling problem. In *Congreso Latino-Iberoamericano de Investigación Operativa*, September 2012.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

J. H. Kingston. Educational timetabling. In A. S. Uyar, E. Ozcan, and N. Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013a. ISBN 978-3-642-39303-7.

J. H. Kingston. The hseval high school timetable evaluator. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi` [Accessed 12/8-2013], Aug. 2013b.

J. H. Kingston. High school timetable file format specification: Constraints. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi?op=spec&part=constraints` [Accessed 12/8-2013], Aug. 2013c.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

M. Pimmer and G. R. Raidl. A timeslot-filling heuristic approach to construct high-school timetables. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Advances in Metaheuristics*, volume 53 of *Operations Research/Computer Science Interfaces Series*, pages 143–157. Springer New York, 2013. ISBN 978-1-4614-6321-4.

G. Post. International timetabling competition 2011 results. `http://www.utwente.nl/ctit/hstt/itc2011/results/` [Accessed 12/8-2013], Aug. 2013a.

G. Post. Benchmarking project for (high) school timetabling. `http://www.utwente.nl/ctit/hstt/` [Accessed 12/8-2013], Aug. 2013b.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

J. Romrös and J. Homberger. An evolutionary algorithm for high school timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 485–488. SINTEF, 2012.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

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

M. Sørensen and T. R. Stidsen. Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. *Annals of Operations Research*, PATAT 2012 SI:Submitted Jan 21. 2013, 2013.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

C. Valouxis, C. Gogos, P. Alefragis, and E. Housos. Decomposing the high school timetable problem. In *Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

Table 6.3: Performance of the MIP on `ALL_INSTANCES`. For each instance is listed the best previously known solution *"Best"*, and for the solution found by our approach is listed the time used to solve Step 1 *"$Time_1$"*, the time used to solve Step 2 *"$Time_2$"*. *"Time"* indicates the total solving time. All times have seconds as unit. Furthermore the objective *"Obj"* and the lower bound *"LB"* is listed. The percentage gap between the objective and the lower bound is divided into the gap for the hard constraints *"Gap_1"* and the gap for the soft constraints, *"Gap_2"*. Objectives in **bold** denote new best solution while optimal solutions are marked with *.

			MIP solution method						
	Instance	Best	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
AU	BGHS98	(3, 494)	>86400	-	>86400	(3, 494)	(-,-)	-	-
AU	SAHS96	(8, 52)	>86400	-	>86400	(8, 52)	(-,-)	-	-
AU	TES99	(1, 140)	>86400	-	>86400	(1, 140)	(0,-)	100.0	-
BR	Instance1	42	0	>86400	>86400	**40**	28	0.0	30.0
BR	Instance2	5	1	>86399	>86400	5	1	0.0	80.0
BR	Instance3	47	1	>86399	>86400	**26**	19	0.0	26.9
BR	Instance4	78	1	>86399	>86400	**61**	42	0.0	31.2
BR	Instance5	43	1	>86399	>86400	**30**	10	0.0	66.7
BR	Instance6	60	1	>86399	>86400	60	14	0.0	76.7
BR	Instance7	122	1	>86399	>86400	122	22	0.0	82.0
DK	Falkoner2012[1]	(2, 23705)	>86400	-	>86400	(2, 23705)	(0,-)	100.0	-
DK	Hasseris2012[2]	(293, 32111)	>86400	-	>86400	(293, 32111)	(-,-)	-	-
DK	Vejen2009[3]	(20, 18966)	>86400	-	>86400	(20, 18966)	(2,-)	90.0	-
UK	StPoul	136	52	>86348	>86400	136	0	0.0	100.0
FI	ElementarySchool	3	2	785	787	*3	3	0.0	0.0
FI	HighSchool	1	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	1	>86399	>86400	88	77	0.0	12.5
GR	UniInstance3[4]	5	0	3	3	*5	5	0.0	0.0
GR	UniInstance4[5]	8	1	>86399	>86400	8	0	0.0	100.0
IT	Instance1	12	1	4561	4562	*12	12	0.0	0.0
IT	Instance4	78	12	>86389	>86400	**62**	27	0.0	56.5
XK[6]	Instance1	3	31	>86369	>86400	3	0	0.0	100.0
NL	GEPRO	(1, 566)	>86400	-	>86400	(1, 566)	(0,-)	100.0	-
NL	Kottenpark2003	1410	57	>86343	>86400	1410	(0,-)	0.0	-
NL	Kottenpark2005	1078	88	>86312	>86400	1078	9	0.0	99.2
NL	Kottenpark2009	9250	92	>86308	>86400	**9035**	160	0.0	98.2
ZA	Woodlands2009	2	22	77878	77900	***0**	0	0.0	0.0
ES	School	(3, 5966)	6525	>79875	>86400	**357**	322	0.0	9.8

[1] Shorthand for instance *FalkonergaardenGymnasium2012*
[2] Shorthand for instance *HasserisGymnasium2012*
[3] Shorthand for instance *VejenGymnasium2009*
[4] Shorthand for instance *WesternGreeceUniversityInstance3*
[5] Shorthand for instance *WesternGreeceUniversityInstance4*
[6] Kosova.

Table 6.4: Performance of the alternative formulation on the smaller instances of archive `ALL_INSTANCES`. All the columns are defined in analogous way to Table 6.3, except for $Obj_{T6.3}$ and $LB_{T6.3}$ which denote the objective value and the lower bound found in Table 6.3.

					MIP alternative formulation						
	Instance	Best	$Obj_{T6.3}$	$LB_{T6.3}$	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
BR	Instance1	42	40	28	0	1918	1918	***38**	38	0.0	0.0
BR	Instance2	5	5	1	0	290	290	*5	5	0.0	0.0
BR	Instance3	47	26	19	1	>86399	>86400	**23**	21	0.0	8.7
BR	Instance4	78	61	42	1	>86399	>86400	61	49	0.0	19.7
BR	Instance5	43	30	10	1	>86399	>86400	**26**	15	0.0	42.3
BR	Instance6	60	60	14	1	>86399	>86400	**59**	18	0.0	69.5
BR	Instance7	122	122	22	1	>86399	>86400	**84**	26	0.0	69.1
FI	HighSchool	1	1	0	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	88	77	1	>86399	>86400	**84**	77	0.0	8.3
GR	UniInstance4[1]	8	8	0	1	>86399	>86400	8	0	0.0	100.0
IT	Instance4	78	62	27	6	>86394	>86400	**57**	27	0.0	52.6
ES	School	(3, 5966)	357	322	44	>86356	>86400	357	330	0.0	7.6

[1] Shorthand for instance *WesternGreeceUniversityInstance4*

Chapter 7 Paper B

Integer Programming and Adaptive Large Neighborhood Search for Real-World Instances of High School Timetabling

Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract A complex model of high school timetabling is presented, which originates from discussions with high schools in Denmark. The model is *complete* in the sense that it contains all relevant practical constraints required by the high schools. Furthermore, the model is used by many institutions to produce their annual timetable. An Integer Programming (IP) formulation of the model is described in detail, and a decomposition approach is briefly discussed. A heuristic based on Adaptive Large Neighborhood Search (ALNS) is also applied. Using 100 real-world datasets, comprehensive computational results are provided which show that the ALNS heuristic outperforms the IP approaches. Furthermore some datasets are made publicly available using the general XHSTT format for high school timetabling.

Keywords High School Timetabling• Modeling• Integer Programming • Adaptive Large Neighborhood Search

7.1 Introduction

The timetabling problem is perhaps the most important problem among the scheduling problems which high schools face. In this paper a complex model of the problem is presented, originating from discussions with high schools in Denmark. Thereby the developed model is tailored to the Danish case of the timetabling problem, and it is *complete* in the sense that it contains all relevant practical constraints required by the high schools. Furthermore, the model is used by many institutions to produce their annual timetable.

This large user base requires a model which is general enough to suit many different requirements, and which is also tractable by computer aided solution methods. This supports the recent trend of developing general models for timetabling problems (see Özcan (2005); Causmaecker and Berghe (2010); Post et al. (2012a); Bonutti et al. (2012)). Furthermore, the timetabling problem of Danish high schools has not been formally described in the literature before. In the remainder of this paper, the problem is denoted as the High School Timetabling Problem (HSTTP).

The HSTTP concerns the construction of a feasible schedule which assigns lectures to timeslots and rooms, and maximizes individual preferences for students and teachers. The problem is in general hard to solve, so an efficient solution approach is important for the high schools. A computer-aided solution approach will allow the high schools to produce more preferable timetables, from the point of view of both teachers, students and the school administration. Three different solution approaches are proposed in this paper, of which two are based on a Mixed-Integer Programming (MIP) model and one is based on Adaptive Large Neighborhood Search (ALNS).

The paper is structured as follows: In Section 7.2 the HSTTP is described in detail, while simultaneously formulating a MIP model. A literature review of related problems and solution approaches is given in Section 7.2.1. Furthermore a decomposition approach of the MIP model is suggested in Section 7.2.3. An ALNS-heuristic is described in Section 7.3. Extensive computational results are presented in Section 7.4. Finally, Section 7.5 concludes on our findings.

7.2 The Timetabling Problem at Danish high schools

In the following the basic timetabling problem at Danish high schools is described, which in Section 7.2.2 is formulated as a MIP model. In Section 7.2.2.2 the MIP is extended by several necessary side-constraints. A detailed formulation of a MIP model has the advantage that each constraint is described in a precise manner. Therefore we postpone the in-depth description of many constraints to Section 7.2.2, and first give an overall description of the problem.

A high school has a number of teachers employed, and has access to rooms where teaching can be performed. Students are taught in classes of different subjects, and each class consists of a number of weekly lectures. This means that we consider classes as an non-physical resource (as opposed to students and teachers), which merely consists of a set of students and a (small) set of teachers. Each day of the week is divided into *modules* where teaching is performed. A combination of a day and a module is denoted a *timeslot*. Students and teachers are preassigned to classes, as we assume the Teacher-Task Assignment problem (Lundberg-Jensen et al. (2008)) and the Student Sectioning problem (Kristiansen and Stidsen (2012)) have already been solved.

Lectures are represented by *events*, and each event must be assigned both a timeslot and a room. However it is also possible to combine several lectures for several classes into the same 'lecture'. This is used when classes should share the same room, e.g. in case of physical education, the sports venue. In such cases, a single event represents several lectures. An event has a set of eligible rooms which it can be assigned to. Often the set of eligible rooms equals the set of all rooms, but some events have special requirements, such as laboratories, sports venue, etc. See Figure 7.1 for an illustration of the notation used.

The timetabling problem essentially consists of creating a schedule for the entire school year, such that events are assigned a timeslot and an eligible room, and such that no clashes among students, teachers or rooms occur. Commonly the problem is solved by assuming that no difference exists among all weeks throughout the year, hence it is sufficient to plan only a single week. The timetable of this week is then replicated for all weeks of the school year, with manual adjustments in case of holidays, illness, etc. In this paper we also consider the case where the school decides to plan two consecutive weeks, which is elaborated in Section 7.2.2.4.

The concept of *EventChains* is introduced in the following. An EventChain forces some events to be in either the same timeslot or in contiguous timeslots. EventChains are very flexible in the sense that they pose no restrictions on which events are chained together. Therefore they can be used to model a lot of special cases required in practice by the high schools. These include, but are not limited to, the following: 1) A double-lecture can be set up by creating

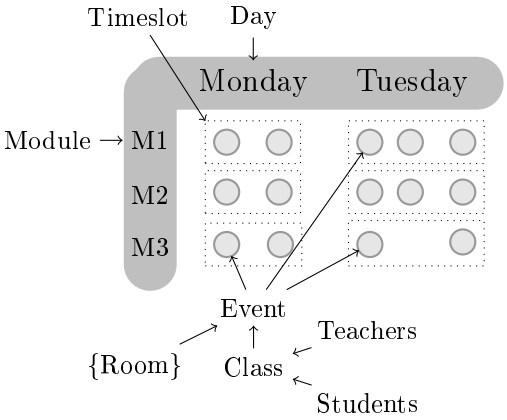


Figure 7.1: Notation used

an EventChain consisting of two events for the same class which must be placed in contiguous timeslots. Similarly, triple-lectures can be created. 2) Parallel double lectures for different classes. 3) Grouping of elective classes in the same timeslot. 4) Large projects where teaching of several classes are combined.

We remark that due to changes in the educational system, the definition of the timetabling problem might change over time. This gives rise to new constraints and changes in the objective weights. What is documented in this paper is how the problem currently looks in the eyes of the high schools, which has proven to be a quite stable formulation of the problem.

7.2.1 Related work

In this section a brief literature review of related work is given.

The definition of Class-Teacher Timetabling (CTT) is more than 50 years old, see Appleby et al. (1961) and Gotlieb (1962). In the original formulation, one is given a set of classes, a set of teachers and a set of periods (timeslots in our terminology). A class is defined as a set of students who follow the exact same curriculum. The goal is to find a schedule where classes meet teachers, fulfilling the teaching demand, subject to no teachers and classes being scheduled more than once in the same period. Furthermore, unavailabilities of teachers and classes in certain periods are given. In these periods, teaching is forbidden. This problem is similar to the basic version of the HSTTP, except rooms are assigned to classes instead of teachers. However, the HSTTP contains many additional constraints, which complicates the problem significantly.

The survey Schmidt and Ströhlein (1980) covers early papers in the area of school timetabling. More recent surveys are Bardadym (1996); Carter and Laporte (1998); Schaerf (1999) and Pillay (2010). The conference series Practice and Theory in Automated Timetabling (PATAT) has contributed largely to the area (see Gendreau and Burke (2008); McCollum et al. (2010); Kjenstad et al. (2012)).

Lawrie (1969) was the first to formulate the basic problem as a column-based MIP. Since, integer programming has been used to model problems with more sophisticated requirements. Some recent contributions include Papoutsis et al. (2003); Avella and Vasil'Ev (2005); Avella et al. (2007); Birbas et al. (2009); Santos et al. (2012). However, integer programming is mainly used for timetabling due to its modeling strength, and not as an actual solution method (Lach and Lübbecke (2012)). With the acknowledgment of modern IP solvers (see Bixby (2012)), this might be undergoing a change. The MIP formulated in this paper is among the most comprehensive models of school timetabling found in the literature.

The International Timetabling Competition 2011 (Post et al. (2012c)) considered a generalized version of high school timetabling (based on the XHSTT format (Post et al. (2012a))). A contribution so far from this competition are several well-performing heuristics, see Fonseca et al. (2012), Kheiri et al. (2012) and Sørensen et al. (2012). Furthermore the competition proved the XHSTT format as a good foundation to build on for future research within the area. In terms of terminology, the model (and therefore also the MIP) presented in this paper is closely related to the XHSTT format. The definitions of events, timeslots, resources (rooms, teachers, students) and classes are equivalent. The most important difference is the fact that the XHSTT format allows events which span multiple timeslots, which our model does not support. Furthermore, HSTTP contains constraints which cannot currently be modeled with XHSTT, see Sørensen and Stidsen (2013) for details. See also Section 7.4.3.

The work presented in this paper has a practical character, as all constraints are the result of requirements made by the high schools. Papers which describe solution methods used in practice are not very common. Both Yoshikawa et al. (1996) and Kingston (2007) describe implementations which are used to create timetables for a few high schools. For universities, both Martin (2004) and Schimmelpfeng and Helber (2007) describe an IP-based approach tailored to a specific university.

7.2.2 Problem Description and a Mixed-Integer Programming Formulation

In this section the HSTTP is described in details, while simultaneously formulation a MIP model. Starting with the basic sets describing the core elements of the problem, Section 7.2.2.1 introduces and motivates the most elementary constraints and variables (the basic model). Sections 7.2.2.2, 7.2.2.3 and 7.2.2.4 extends the basic model w.r.t. requirements made by the high schools, and introduces a variety of soft constraints. All in all, this builds a model of the HSTTP step-by-step, introducing parameters and auxiliary variables in the MIP as we need them.

The presented model of HSTTP is stated such that a feasible solution always exists, as it is possible to *not* assign an event to a timeslot and/or a room. However in such cases a large penalty is given. This formulation of the model has been agreed upon during our discussions with the high schools, and resembles the fact that a solution to the model should be a *satisfactory* timetable, which is *partial* if necessary. A partial timetable is defined as one where not all events are assigned a timeslot and a room. A satisfactory timetable (which is possibly partial) is one which fulfills all the requirements (hard and soft constraints) desired by the high schools, and can be used as-is. Hence the timetables provided by our model are of high quality, but might require additional work by the respective high school if a timetable is partial. On the other hand, if a solution to the model is not partial, then it is guaranteed that the solution represents desirable timetables for all stakeholders. Thereby the model should be used in a decision-support context by the high schools, possibly performing subsequent runs of the solution algorithm with different input criteria (various parameters w.r.t. characteristics of the desired timetables can be provided). How we deal with events not assigned a timeslot is discussed under future research in Section 7.5.

The model of the HSTTP is described in the following. The following sets are given: Days $\mathcal{D}$, modules $\mathcal{M}$, timeslots $\mathcal{T}$, events $\mathcal{E}$, and rooms $\mathcal{R}$. The set of entities is denoted $\mathcal{A}$, which includes both students and teachers. This means that an entity $a \in \mathcal{A}$ is either a student or a teacher. Grouping these two types of entities in the same set allows us to simplify notation for certain constraints. To reduce problem size, students which are assigned exactly the same events are grouped into one super-entity. This is related to the concept of *curricula* in university course timetabling, and reduces the problem size considerably. Let $M_a \in \mathbb{N}$ denote the number

of 'real' entities which entity $a \in \mathcal{A}$ represents (so $M_a = 1$ if entity a is a teacher). As discussed in Section 7.2, a class is defined as a set of entities which are taught/teaches a specific subject. Each class is then associated with a certain number of events, representing the lectures which should be scheduled throughout the week. The set of classes is denoted $\mathcal{C}$. In addition, let $\rho(i)$ denote the zero-based ordinal number of $i \in I$, e.g. if $I = \{a, b, c\}$, then $\rho(b) = 1$ with respect to set I.

Below constraints and objective function-terms of the problem are stated. First a basic model with well known constraints is introduced, and afterwards expanded to allow more specialized constraints. Finally various *soft constraints* are added, which models different quality-metrics of the timetable. The notation used is lazy; $\forall e$ is short for $\forall e \in \mathcal{E}$, $\forall e \neq e'$ is short for $\forall e \in \{\mathcal{E} \setminus \{e'\}\}$, and $\sum_e$ is short for $\sum_{e \in \mathcal{E}}$. Parameters are written in uppercase (except for cost-parameters in the objective which are denoted with Greek letters), and variables are written in lowercase.

7.2.2.1 Basic model

The main decision variable is $x_{e,r,t} \in \{0, 1\}$, which takes value 1 if event e is scheduled to room r and timeslot t, and 0 otherwise. Each event should be assigned exactly one room and one timeslot, so we introduce the constraint

$$\sum_{r,t} x_{e,r,t} = 1 \qquad \forall e \tag{7.1}$$

To ensure a feasible solution exists, the set of timeslots and the set of rooms are extended by 'dummy' elements, which models that an event is *not* assigned a timeslot or a room, respectively. This means that $\mathcal{T} = \{\mathcal{T} \cup \{t_D\}\}$ and $\mathcal{R} = \{\mathcal{R} \cup \{r_D\}\}$. An assigning to either of these dummy-elements yields a big penalty (defined in Section 7.2.4).

The auxiliary variable $y_{e,t} \in \{0, 1\}$ is introduced, which takes value 1 if event e is placed in timeslot t, and is constrained by the following

$$\sum_{r} x_{e,r,t} = y_{e,t} \qquad \forall e, t \tag{7.2}$$

This auxiliary variable is introduced for two reasons; 1) It simplifies the notation of many of the constraints introduced further on. 2) It greatly reduces the number of non-zeros in the model, which reduces the memory-consumption when solving the MIP model.

Each entity can only participate in one event in each timeslot, except in the dummy-timeslot. Let $B_{e,a} \in \{0, 1\}$ take value 1 if entity a is part of event e, and 0 otherwise. The following constraint is imposed,

$$\sum_{e} B_{e,a} y_{e,t} \leq 1 \qquad \forall a, t \neq t_D \tag{7.3}$$

Each room (except for the dummy room) can only be assigned once to each timeslot (except in the dummy timeslot). Furthermore, a room might be unavailable for teaching in certain time slots, for instance if it is shared by the high school and other institutions, or if it is undergoing maintenance, etc. Let $G_{r,t} \in \{0, 1\}$ take value 1 if room r is available at timeslot t, and 0 otherwise. This constitutes the following constraint,

$$\sum_{e} x_{e,r,t} \leq G_{r,t} \qquad \forall r \neq r_D, t \neq t_D \tag{7.4}$$

An event might be locked to a specific timeslot or a specific room. Let $LT_{e,t} \in \{0,1\}$ take value 1 if event e is locked to timeslot t, and let $LR_{e,r} \in \{0,1\}$ take value 1 if event e is locked to room r. The following constraints are imposed,

$$y_{e,t} = 1 \qquad \forall e, t, LT_{e,t} = 1 \tag{7.5}$$

$$\sum_t x_{e,r,t} = 1 \qquad \forall e, r, LR_{e,r} = 1 \tag{7.6}$$

Some events might require special rooms, i.e. chemistry lectures or physical education. Let $K_{e,r} \in \{0,1\}$ take value 1 if event e can be assigned to room r, and 0 otherwise. The following constraint is imposed,

$$\sum_t x_{e,r,t} \leq K_{e,r} \qquad \forall e, r \tag{7.7}$$

It is not possible to assign a room to an event unless the event is also assigned a timeslot. This is because assigning timeslots is considered far more important than assigning rooms, and therefore it does not make sense to assign a room unless the event has a timeslot. Consider for instance an event assigned to the dummy timeslot. This event can be assigned to any room without violating the room conflict constraint (7.4). On the other hand, it is completely legal to assign an event to a timeslot, but not to a room. The following constraint is imposed,

$$\sum_{r \in \mathcal{R} \setminus \{r_D\}} x_{e,r,t_D} - \sum_r LR_{e,r} \leq 0 \qquad \forall e \tag{7.8}$$

This constraint specifies that if event e is not locked to any room, then it cannot be assigned to both a room different from the dummy-room r_D and the dummy-timeslot t_D. However if the event is locked to a room, it is legal to assign it to this room and the dummy-timeslot.

If an event is not assigned to a timeslot and/or a room, a penalty must be imposed. Furthermore, various other criteria define priorities for the assignment of events to timeslots/rooms, which we elaborate on in Section 7.2.4. Denote this penalty by $\alpha_{e,r,t} \in \mathbb{R}^+$. The objective of the model therefore reads,

$$\min \sum_{e,r,t} \alpha_{e,r,t} x_{e,r,t} \tag{7.9}$$

7.2.2.2 Extended model

In this section, the model is extended to allow for constraints which arise from the concept of EventChains and other features required in practice by the high schools.

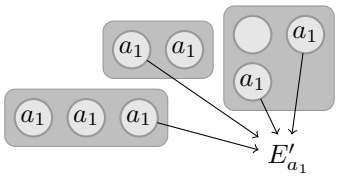
The EventChains are modeled by specifying that some events should be assigned the same timeslot as others, and that some events should be in contiguous timeslots. Let $S_e \subseteq \mathcal{E}$ be the set of events which must be assigned the same timeslot as event e, and let $C_e \subseteq \mathcal{E}$ be the set of events which must be assigned the timeslot following immediately after the timeslot assigned to event e. Furthermore, d_t denotes the day $d \in \mathcal{D}$ of timeslot t. The following two constraint are added to the model,

$$y_{e,t} - y_{e',t} = 0 \qquad \forall e, e' \in S_e, t \tag{7.10}$$

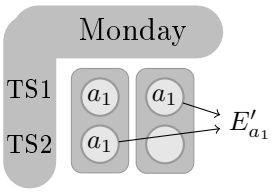
$$y_{e,t} - y_{e',t'} = 0 \qquad \forall e, e' \in C_e, t, t', d_t = d_{t'}, \rho(t) + 1 = \rho(t') \tag{7.11}$$

The EventChains imply further restrictions; Since an EventChain can be created which in itself causes a conflict between entities in terms of constraints (7.3) and (7.10) (i.e. more than

one event containing the same entity should be scheduled to the same timeslot), modifications to constraint (7.3) are needed. As we can a priori determine between which events in some EventChain an entity conflict will occur, we simply exclude some events from constraint (7.3). Let the set $E'_a \subseteq \mathcal{E}$ be the subset of events for which entity conflicts are checked for entity a. See Figure 7.2.



(a) Three EventChains with several events where entity a_1 participates at same offset. Only a subset of events are checked for conflicts for entity a_1.



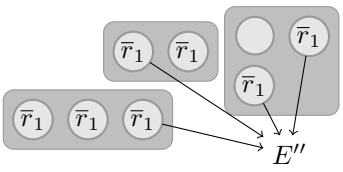
(b) Two EventChains locked to timeslots, with several events where entity a_1 participates. Only a subset of events are checked for conflicts for entity a_1.

Figure 7.2: Only some events are checked for entity conflicts

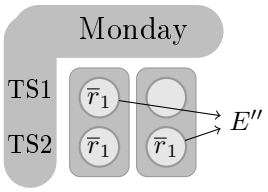
The modified version of constraint (7.3) is shown below,

$$\sum_{e \in E'_a} y_{e,t} \leq 1 \qquad \forall a, t \in \mathcal{T} \setminus \{t_D\} \tag{7.3'}$$

Furthermore, EventChains might also cause conflicts for a room in terms of constraint (7.4), if several events are locked to the same room. See Figure 7.3. We introduce the set E'', denoting



(a) Three EventChains with several events locked to room r_1 for same offset. Only a subset of these events are checked for conflicts.



(b) Two EventChains locked to timeslots, with several events locked to room r_1. Only a subset of these events are checked for conflicts.

Figure 7.3: Room conflicts exceptions

the events which should be checked for room conflicts. Constraint (7.4) is modified to read

$$\sum_{e \in E''} x_{e,r,t} \leq G_{r,t} \qquad \forall r \neq r_D, t \neq t_D \tag{7.4'}$$

7.2.2.3 Timetable quality metrics

In this section, the model is extended by several quality metrics for a timetable. Most of these are modeled in the form of unwanted properties, whose (weighted) quantitative appearance should be minimized in the objective function, commonly known as soft-constraints. However also a few hard-constraints are described, as some properties of a timetable are considered infeasible.

Idle timeslots An undesirable property of a timetable for an entity is *idle* timeslots. An idle timeslot for an entity has no events scheduled, but there is both an earlier and a later timeslot on that day where an event is scheduled, hence the entity must sit idle throughout this timeslot. By interviewing the high schools, it is our experience that they especially consider idle timeslots for students to be undesirable. This is typically due to 1) A student typically participates in so many events that a completely compact timetable seems to be possible, and/or 2) The high school believes that students are unlikely to do school-related tasks in an idle timeslot. For teachers, idle slots are also undesirable. However the high school will much prefer an idle timeslot for a teacher over an idle timeslot for a student.

Let the variable $h_{a,d} \in \mathbb{N}_0$ be the number of idle timeslots for entity a on day d, and let $\beta_a \in \mathbb{R}^+$ be the cost of an idle timeslot for entity a. The objective function term for this constraint is given by

$$\sum_{a,d} \beta_a h_{a,d} \tag{7.12}$$

Let the variables $\underline{h}_{a,d} \in \mathbb{N}_0$ and $\overline{h}_{a,d} \in \mathbb{N}_0$ be ordinal number of the first and last timeslot where entity a is active on day d, respectively. The following constraints are imposed,

$$\overline{h}_{a,d} - \underline{h}_{a,d} - \sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} + 1 = h_{a,d} \qquad \forall a, d \tag{7.13}$$

$$|\mathcal{M}| - (|\mathcal{M}| - \rho(t)) \sum_{e \in E'_a} y_{e,t} \geq \underline{h}_{a,d} \qquad \forall a, d, t \in \mathcal{T}_d \tag{7.14}$$

$$\rho(t) \sum_{e \in E'_a} y_{e,t} \leq \overline{h}_{a,d} \qquad \forall a, d, t \in \mathcal{T}_d \tag{7.15}$$

Equations (7.14) and (7.15) ensures that variables $\underline{h}_{a,d}$ and $\overline{h}_{a,d}$ are constrained properly. In case entity a has no activities on day d (which naturally entails no idle timeslots), the value of $\underline{h}_{a,d}$ can be set to 1, to avoid $h_{a,d}$ to take value 1. Notice that these constraints use big-M notation, which is known to give bad LP-relaxations. This might have negative impact on solution times. A formulation without big-M notation is known, but it requires too many extra constraints to be applicable.

Unavailabilities For each event, it might be infeasible to assign it to certain times. This is used to prohibit teaching of certain classes at certain times. For instance it is common that teaching of first year students is undesirable in the late modules on each day. Another example is to prohibit all teaching in the last module on Fridays, and only use this module in case a solution without it cannot be found.

Let $D_{e,t} \in \{0, 1\}$ take value 1 if it is feasible to assign event e to timeslot t. The following constraint for event unavailability is imposed,

$$\sum_{t, D_{e,t}=0} y_{e,t} = 0 \qquad \forall e \tag{7.16}$$

Furthermore, certain timeslots are undesirable for certain teachers. These 'soft-unavailabilities' are handled by simply adjusting the weight $\alpha_{e,r,t}$ for these timeslots for those events which the teacher is part of, see Section 7.2.4.

Days off It is quite common to require that all teachers have at least one day off, i.e. a day without any scheduled events. They can for instance use this day for preparation of future lectures. Let $F_a \in \mathbb{N}_0$ be the number of days off required for entity a (takes value 0 for all students). It is the job of the solution method to decide which days should have a no scheduled events. Let $f_{a,d} \in \{0,1\}$ take value 1 if entity a has no events on day d, and 0 otherwise. To make this variable take appropriate values, it is incorporated in constraint (7.3′). The rephrase of constraint (7.3′) is denoted (7.3″), which constraints the problem in equivalent way, but also makes $f_{a,d}$ take appropriate values,

$$\sum_{e \in E'_a} y_{e,t} + f_{a,d} \leq 1 \qquad \forall a, d, t \in \mathcal{T}_d \tag{7.3''}$$

The following constraint ensures entities are assigned to their required number of days off,

$$\sum_{d} f_{a,d} \geq F_a \qquad \forall a \tag{7.17}$$

Besides the required number of days off, it is generally preferred for teachers to have as many days off as possible. Therefore we also maximize the number of days off in the objective,

$$\sum_{a} \gamma_a \left(|\mathcal{D}| - \sum_{d} f_{a,d} \right) \tag{7.18}$$

where $\gamma_a \in \mathbb{R}^+$ denotes the penalty for a day not being a day-off for entity a. Notice that the cardinality of $\mathcal{D}$ is incorporated to avoid this term to go below 0. Thereby the lower bound of the entire MIP is kept at 0.

The high schools prefer that students have no days off. Therefore days off for students are penalized by adding the following term to the objective,

$$\sum_{a,d} \delta_a f_{a,d} \tag{7.19}$$

where $\delta_a \in \mathbb{R}^+$ is the penalty for a entity a having a day off (takes value 0 for all teachers). Notice that since this expression minimizes $f_{a,d}$, constraint (7.3″) does not constrain $f_{a,d}$ sufficiently. Constraint (7.3″) only specifies that if an entity a has at least one event on day d, $f_{a,d}$ must take value 0. This means that if an entity a has no events assigned on some day d, we need to make sure $f_{a,d}$ is forced to take value 1. This is done by the following constraint,

$$\sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} + f_{a,d} \geq 1 \qquad \forall a, d \tag{7.20}$$

Room stability It is important for teachers and students of a class that all lectures of this particular class take place in the same room. We therefore aim at minimizing the number of different rooms assigned to events where a given class participates. Let $v_{c,r} \in \{0,1\}$ take value 1 if class c is assigned to room r at least once, and 0 otherwise. Let $J_{e,c} \in \{0,1\}$ take value 1 if class c is part of event e, and 0 otherwise. The following constraint is imposed,

$$\sum_{e, t \neq t_D} J_{e,c} x_{e,r,t} - \sum_{e} J_{e,c} v_{c,r} \leq 0 \qquad \forall c, r \tag{7.21}$$

Let $s_c \in \mathbb{N}_0$ be the number of rooms assigned to class c minus one, i.e. the number of 'excess' rooms. This variable is constrained by:

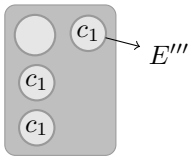
$$\sum_r v_{c,r} - 1 \leq s_c \qquad \forall c \tag{7.22}$$

$\epsilon \in \mathbb{R}^+$ denotes the cost of each excess room assigned to class c. The following term is added to the objective function,

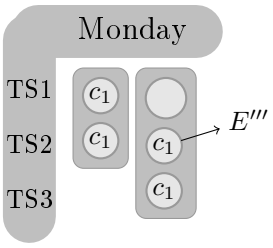
$$\epsilon \sum_c s_c \tag{7.23}$$

Day-conflicts Each class can only be taught once each day, unless several events containing the same class are part of the same EventChain (e.g. double-lectures), or unless several events of this class are locked to timeslots on this day. Let variable $b_{c,t} \in \{0, 1\}$ take value 1 if class c is part of at least one event on day d, and let $E''' \subseteq \mathcal{E}$ be the set of events for which day-conflicts are checked. All events are included in E''', with the following exceptions (see also Figure 7.4):

- Some events of the same class are part of the same EventChain. All of these, except one, is excluded from E'''.
- If multiple events are locked to timeslots on the same day, all of these events, except one, are excluded from E'''.



(a) One EventChain where class c_1 participates in several events. Only one of these events is added to E'''.



(b) Two EventChains locked to the same day. Only one event which c_1 is part of is checked for day-conflicts.

Figure 7.4: Day conflicts exceptions

The following constraint is added to the model,

$$\sum_{e \in E'''} J_{e,c} y_{e,t} \leq b_{c,t} \qquad \forall c, t \tag{7.24}$$

Day-conflicts of classes are thereby avoided by adding following constraint,

$$\sum_{t \in \mathcal{T}_d} b_{c,t} \leq 1 \qquad \forall c, d \tag{7.25}$$

This constraint could be stated without variable $b_{c,t}$, but this variable is used in other constraints as well.

Neighbor days for classes Another undesirable property for a timetable is neighbor-day-clashes for classes. Both students and teachers prefer that lectures of a class are spread throughout the week, for instance to allow more time for homework between lectures. Let $P_{d,d'}$ take the value 1 if day d and day d' are neighbor days, and 0 otherwise. Neighbor-day pairs are Monday-Tuesday, Tuesday-Wednesday, etc., excluding Tuesday-Monday, Wednesday-Tuesday, etc. Let the variable $n_{c,d} \in \{0,1\}$ take value 1 if class c has a neighbor-day-conflict on day d, and 0 otherwise. If class c is locked to at least one event on two contiguous days, this is not defined as a conflict. Let $R_{c,d} \in \{0,1\}$ take value 1 if class c is locked to some event on day d, and 0 otherwise. The following constraints are imposed,

$$\sum_{t \in \mathcal{T}_d} b_{c,t} + \sum_{t \in \mathcal{T}_{d'}} b_{c,t} - n_{c,d} \leq 1 \qquad \forall c, d, d', P_{d,d'} = 1, R_{c,d} + R_{c,d'} \leq 1 \tag{7.26}$$

Neighbor-day conflicts are penalized by the following term in the objective (where $\zeta \in \mathbb{R}^+$),

$$\zeta \sum_{c,d} n_{c,d} \tag{7.27}$$

In case a class has few lectures, neighbor-day conflicts might even be infeasible. Let $N_c \in \mathbb{N}_0$ be the number of allowed neighborday-conflicts for class c, defined as follows:

$$N_c = \begin{cases} 0 & NC_c \leq 2 \\ w & NC_c = 3 \\ 3w & NC_c = 4 \\ 4w & NC_c \geq 5 \end{cases} \tag{7.28}$$

where NC_c is the number of EventChains where class c participates, and $w \in \{1,2\}$ is the number of weeks being planned. The following constraint is added to the model,

$$\sum_{d} n_{c,d} \leq N_c \qquad \forall c \tag{7.29}$$

Teacher daily workload It can be preferred for teachers that they do not have to teach in all modules on a day. Let $W_a \in \mathbb{N}_0$ be the maximum number of lectures on a day for entity a (takes value 0 for all student entities). The following constraint is imposed,

$$\sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} \leq W_a \qquad \forall a, d \tag{7.30}$$

On the other hand, teachers do not like days with too few lectures. It is generally believed among the high schools that a teacher should have at least two lectures on 'active' days, i.e. days with only one lecture are undesirable. In the following the model is constrained so days with only one active timeslot for an entity is penalized. Let $o_{a,d} \in \{0,1\}$ take value 1 if entity a is a teacher and has only one lecture on day d, and 0 otherwise. The cost-parameter for such days is $\zeta \in \mathbb{R}^+$. The following constraint is imposed,

$$2 - \sum_{e \in E'_a, t \in \mathcal{T}_d} y_{e,t} - 2f_{a,d} \leq o_{a,d} \qquad \forall a, d \tag{7.31}$$

The following expression is added to the objective,

$$\sum_{a,d} \eta_a o_{a,d} \tag{7.32}$$

7.2.2.4 Two week schedule metrics

Planning two weeks instead of one gives twice the amount of timeslots, and thereby larger flexibility, which is prefered by some high schools. For instance suppose a class on average should have five lectures each week. Instead of assigning five events to each week, four events could be assigned to the first week, and six events could be assigned to the second week. The planning of two weeks yields additional quality metrics.

Days off stability for teachers It is preferred to have the required days off for entities distributed equivalently among the two weeks, e.g. in case of 3 required days off, each week must contain at least 1 day off. This is done by the following constraint,

$$\Big| \sum_{d \in d(\underline{\mathcal{T}})} f_{a,d} - \sum_{d \in d(\overline{\mathcal{T}})} f_{a,d} \Big| \leq 1 \qquad \forall a \tag{7.33}$$

where $d(\underline{\mathcal{T}})$ and $d(\overline{\mathcal{T}})$ denotes days of the first and second week, respectively. This constraint is easily transfered into linear-form using two sets of constraints by the following approach. Consider the expression $|Y_1 - Y_2| \leq 1$ for which we desire a linear form. This is achieved by introducing the constraints $Y_1 - Y_2 \leq 1$ and $Y_2 - Y_1 \leq 1$.

Stability for lectures of classes Likewise, the distribution of lectures for classes should also be evenly distributed between weeks. Let $w_c \in \mathbb{N}_0$ be the number of events out of week-balance for class c. This is punished in the objective by (where $\iota \in \mathbb{R}^+$)

$$\iota \sum_c w_c \tag{7.34}$$

and is constrained by the following,

$$\Big| \sum_{e,t \in \underline{\mathcal{T}}} J_{e,c} y_{e,t} - \sum_{e,t \in \overline{\mathcal{T}}} J_{e,c} y_{e,t} \Big| - 1 = w_c \qquad \forall c \tag{7.35}$$

This expression can also be linearized using the described approach.

7.2.2.5 The Full Model

All ingredients of the MIP model has now been described. The objective of the model is to minimize the sum of eqs. (7.9), (7.12), (7.18), (7.19), (7.23), (7.27), (7.32), (7.34), subject to the described bounds on the variables and eqs. (7.1)-(7.2), (7.3″), (7.4′), (7.5)-(7.8), (7.10)-(7.11), (7.13)-(7.17), (7.20)-(7.22), (7.24)-(7.26), (7.29)-(7.31), (7.33), (7.35). Notice that all variables except for $x_{e,r,t}$, $y_{e,t}$ and $v_{c,r}$ can be stated as LP variables, as they will naturally take integer values. It is expected that not having integer requirements on these variables will facilitate a more efficient solution procedure. Furthermore, a formal proof of $\mathcal{NP}$-hard based on this model can be found in Sørensen and Stidsen (2013).

7.2.3 Two-Stage Decomposition

Inspired by the approach taking in Lach and Lübbecke (2012), we propose to solve the MIP model in two stages. In stage one, events are assigned to timeslots, and in stage two, events are assigned to rooms given their assigned timeslots. By this approach, the explosion in the number of variables caused by $x_{e,r,t}$ is avoided, as each stage can instead be modeled by a binary variable with two indices.

7.2.3.1 Stage One

In stage one, set $\mathcal{R} = \{\mathcal{R}_L \cup r_D\}$, where $\mathcal{R}_L$ is the set of rooms which are locked to at least one event. If an event e is not locked to a room, set the dummy-room as the only feasible room for this event, i.e. $K_{e,r_D} = 1$ and $K_{e,r} = 0\ \forall r \neq r_D$. This forces all events which are not locked to a room to be assigned to the dummy-room. By this setting of parameters, exactly one feasible room exists for each event, which significantly reduces the number of variables in terms of $x_{e,r,t}$. This means that $x_{e,r,t}$ can be substituted by $K_{e,r}y_{e,t}$.

As we would like to not only generate good solutions by this approach, but also to generate lower bounds, it is assumed that each event can be assigned the best room possible. Hence the $\alpha_{e,r,t}$ parameter is temporarily modified by

$$\alpha_{e,r,t} = \min_{r'} \alpha_{e,r',t} \tag{7.36}$$

Furthermore, the room stability constraints (7.21) and (7.22) are removed. These constraints are not redundant as they still apply to all locked rooms, but the constraints must be removed to generate a valid lower bound, in the sense that some of the penalty produced by these constraints might disappear when additional rooms are assigned in the stage two model.

With these modifications, the MIP model is solved to obtain a solution $y^*_{e,t}$ where events are assigned timeslots. The lower bound obtained by solving this model is a lower bound on the original MIP. Notice that no constraints are imposed to ensure events can be assigned an eligible room in the next stage. This might give us worse solutions, but it is expected that the natural spread among the timeslots events are assigned to will also ensure a fair amount of rooms can be assigned without causing conflicts. It is expected that not adding additional constraints will give an easier model to solve.

7.2.3.2 Stage Two

In stage two, the MIP is solved with the variables $y_{e,t}$ fixed as set by $y^*_{e,t}$. This turns the problem into a matter of assigning rooms to events, and this problem has a lot less variables than the original problem. All constraints are redundant, except for (7.4′), (7.21) and (7.22). A feasible solution for this model is clearly a feasible solution to the original MIP.

7.2.4 Weights

Below are listed the values of the weights used in the objective. These values have been selected on the basis of experiments performed in cooperation with the high schools. A number of high schools have chosen a set of parameter values which they felt produced good timetables, and the numbers reported here are the averages. The individual preferences of each high school can easily be taken into account by allowing them to adjust the weights in the input given to the solution algorithms.

The value of $\alpha_{e,r,t}$ has the most complex definition, as it models several different requirements. These are as follows:

- Events locked to timeslots and/or rooms receive no penalty for these respective assignments.
- The penalty of the dummy-timeslot and dummy-room are high.
- Each teacher has a set of timeslots which are undesirable (possibly empty). Let $V_{a,t}$ takes value 1 if it is undesirable for entity a to be assigned timeslot t (takes value 0 for all student entities).

- 'Early' timeslot are generally more preferable than 'late' timeslots. Let m_t denote the module of timeslot t.
- An event has preferences for the room it is assigned. This is defined by $p_{e,r} \in \{1,2,3\}$, which is the priority of room r for event e.
- The high schools prefer that consecutive runs of the algorithms produce similar solutions. Therefore a small penalty is imposed on assignments of timeslots and rooms which deviate from those of the previous solution. Let $t_P(e)$ and $r_P(e)$ be the previous timeslot and previous room assigned to event e, respectively. If no previous solution exists, these values are omitted.

$$\alpha_{e,r,t} = \begin{cases} 0 & LT_{e,t} = 1 \\ 60 & t = t_D \\ 2\rho\left(m_t\right) + 4\sum_a M_a B_{e,a} V_{a,t} & \text{else} \end{cases} + \begin{cases} 0 & LR_{e,r} = 1 \\ 10 & r = r_D \\ 2(p_{e,r}-1) & \text{else} \end{cases} + \begin{cases} 1 & t = t_P(e) \\ 0 & \text{else} \end{cases} + \begin{cases} 1 & r = r_P(e) \\ 0 & \text{else} \end{cases} \tag{7.37}$$

Remaining weights are defined as follows:

Idle slots:	$\beta_a = 6M_a$ if a is teacher, $7M_a$ if a is student	(7.38)
Teachers days off:	$\gamma_a = M_a$ if a is teacher, 0 if a is student	(7.39)
Students days off:	$\delta_a = 0$ if a is teacher, M_a if a is student	(7.40)
Room stability:	$\epsilon = 1$	(7.41)
Class neighbor-days:	$\zeta = 8$	(7.42)
Entity only one lecture:	$\eta_a = 4M_a$ if a is teacher, 0 if a is student	(7.43)
Week imbalance for class:	$\iota = 8$	(7.44)

Notice that by far the biggest penalty comes from not assigning an event to a timeslot.

7.3 Adaptive Large Neighborhood Search

In this section a heuristic solution approach based on Adaptive Large Neighborhood Search (ALNS) is described.

Adaptive Large Neighborhood Search is a recent extension of the Large Neighborhood Search (LNS) paradigm, often credited to Ropke and Pisinger (2006). As in the LNS framework, first a destruct (ruin/remove) operator is applied to the solution at hand, and then a construct (recreate/insert) operator is used to repair the solution. In an ALNS framework, multiple destruct and construct operators are used, and the adaptive layer keeps track of their individual performance, and increases the probability of selecting operators which have previously performed 'well'. ALNS has mainly been applied to variants of the Vehicle Routing Problem (VRP) (Azi et al. (2010); Hemmelmayr et al. (2011); Salazar-Aguilar et al. (2011); Ribeiro and Laporte (2012)), but lately also other problem-domains (Muller et al. (2011); Muller (2010)). This particular implementation of ALNS is similar in structure to that of Sørensen et al. (2012) and Kristiansen et al. (2013). Details are given in the following.

An acceptance criteria of new solutions is borrowed from the *Simulated Annealing* meta-heuristic. This means that worse solutions have a certain probability to be selected as the new incumbent solution S_{cur}, and this probability is decreased as the algorithm progresses. Let $T \in \mathbb{R}^+$ denote the current *temperature* and denote by $z(S)$ the objective value of solution S. By description of LNS, one new solution S_{new} is found in each iteration. This solution is accepted with probability $\exp\left(\frac{z(S_{\text{cur}})-z(S_{\text{new}})}{T}\right)$.

The initial temperature T_0 is selected such that a solution with an objective which is $w_{\text{SA}} \in]0;1[$ percent worse than the initial solution S_0 is selected with probability 0.5, i.e. $T_0 = \frac{w_{\text{SA}} \cdot z(S_0)}{\ln 2}$. In each iteration the temperature is decreased by the decay factor $d_{\text{SA}} \in]0;1[$, using the equation $T = d_{\text{SA}} T$.

The set of repair and destroy methods are denoted Ω^+ and Ω^-, respectively. In the following *method* is used to denote both a repair and a destroy method. A run of the algorithm is divided into segments $\{t_0, t_1, \ldots, t_n\}$ each consisting of N_{it} iterations. Let π_i^t be the weight of method i in segment t. Initially in the first section t_0, $\pi_i^{t_0} = 1 \; \forall i$. The probability of choosing method i in segment t is $\frac{\pi_i^t}{\sum_j \pi_j^t}$. At the end of each segment t, the following update is performed for all methods,

$$\pi_i^{t+1} = \rho \frac{\bar{\pi}_i^t}{a_i^t} + (1-\rho)\pi_i^t \tag{7.45}$$

where a_i^t is the number of times method i has been selected in segment t. $\bar{\pi}_i^t$ is the *observed* weight of method i in segment t, which in each iteration is incremented depending on the quality of the new found solution. $\rho \in [0,1]$ is the reaction factor. A high reaction factor means that the weights of a segment will be very dependent upon the observed weights of the previous segment.

The observed weight $\bar{\pi}_i^t$ is updated in each iteration, by the following:

$$\text{gap} = \frac{z(S_{\text{cur}}) - z(S_{\text{new}})}{z(S_{\text{cur}})} \tag{7.46}$$

$$\bar{\pi}_i^t = \bar{\pi}_i^t + 5^{\min(\sigma \cdot \text{gap}, 1)} \tag{7.47}$$

where $\sigma \in \mathbb{R}^+$ is the scaling parameter.

Hence this ALNS algorithm contains parameters w_{SA}, d_{SA} for controlling the acceptance-criteria, and parameters N_{it}, ρ, σ for controlling the adaptive selection of insert/remove methods.

A solution to the HSTTP is a list of (event,room,timeslot)-tuples, each representing an assignment of an event to a room and to a timeslot. The concept of a *move* is defined as follows: Given some feasible solution to an instance of the HSTTP, the move M permutes the solution S_1 such that a new feasible solution S_2 is obtained, and the change in the objective is $\Delta(M) = z(S_2) - z(S_1)$. For simplifying notation we only consider moves which do not yield an infeasible solution, however in practice such moves exist, but since they will never be applied to the solution in our implementation, they are ignored in this description. Four classes of moves have been implemented:

- $M_{ec,t}^{\text{time}}$ assigns EventChain ec to timeslot t.
- $\not{M}_{ec,t}^{\text{time}}$ un-assigns EventChain ec from timeslot t.
- $M_{e,r}^{\text{room}}$ assigns event e to room r.
- $\not{M}_{e,r}^{\text{room}}$ un-assigns event e from room r.

As the assign-time moves apply to EventChains, as opposed to events, the constraints for events which should be placed in the same/contiguous timeslots are handled implicitly, which simplifies the implementation. By these moves, the remove- and insertion-operators are constructed.

7.3.1 Insertion methods

Algorithm 1 shows the general pseudo-code for the implemented insertion methods. M^{time} denotes a move which assigns an EventChain to a timeslot. The insertion methods differ in how they select this move in each iteration (line 3 in the algorithm). Once an assign-time move has been selected, it is attempted to perform an assign-room move on each of the events in the EventChain of this assign-time move. This can possible be improved by future research, e.g. by the approach of Kostuch (2005). In the following, the approach for selecting the assign-time move is described for each insertion-method.

Algorithm 1 Insertion method

```
1:  input: A feasible solution S
2:  loop
3:      M^time = ...
4:      if Δ(M^time) > 0 then stop
5:      apply M^time to S
6:      for all e ∈ ec(M^time) do
7:          M^room = arg min_r Δ(M^room_{e,r})
8:          if Δ(M^room) < 0 then apply M^room to S
9:      end for
10: end loop
```

7.3.1.1 InsertGreedy

This method iteratively performs the assign-time move which reduces the objective most, i.e. $M^{\text{time}} = \arg\min_{ec,t} \Delta\left(M^{\text{time}}_{ec,t}\right)$, until the delta-value of the best move is positive (i.e. the best move makes the solution worse).

7.3.1.2 InsertRegret-k

This is similar to the Regret-N neighborhood applied to variants of the VRP (Tillman and Cain (1972); Martello and Toth (1981); Potvin and Rousseau (1993)). Let $k \in \{2, 3, \ldots, |\mathcal{T}|\}$, and let $M^{\text{time}}_{ec,\{i\}}$ denote the i-th best time-move for EventChain ec. For a given k, the move selection is given by:

$$M^{\text{time}} = \underset{\substack{ec \\ \Delta\left(M^{\text{time}}_{ec,\{1\}}\right)<0}}{\arg\min} \left(\Delta\left(M^{\text{time}}_{ec,\{1\}}\right) - \sum_{i \geq 2}^{k} \Delta\left(M^{\text{time}}_{ec,\{i\}}\right) \right) \tag{7.48}$$

To explain this equation, we consider an InsertRegret-2 method as an example. This method selects in each iteration the best time move for the EventChain where the difference between the best time move and the second-best time move is most negative. The intuition is to perform the move which we will regret most if not done now. The following choices of k have been made by basic tests: 2, 3, 4, $|\mathcal{T}|$, which constitute four different insertion methods.

7.3.2 Remove methods

In each iteration of the ALNS, a number of events $q \in \mathbb{N}$ is selected to be unassigned from timeslots in the solution at hand. The quantity q is selected as a random integer in the interval $[3, \max(p|\mathcal{E}|, 5)]$, where the parameter $p \in [0; 1]$ describes the maximum percentage of events to

be removed. As in Kristiansen et al. (2013), p is decreased with time, i.e. at time 0: $p = p^{\text{des}}_{\text{start}}$, and when reaching the timelimit: $p = p^{\text{des}}_{\text{end}}$. In between, a linear decay of the parameter is applied.

Each remove-method contains its own specific remove procedure. The general concept of all remove methods considered are the following: Repeat the remove procedure, until q events have been unassigned from their respective timeslot/room.

Recall that an event cannot be assigned a room if it is not assigned a timeslot (constraint (7.8) in the MIP). Throughout this section it is therefore implicitly handled, that if an event is unassigned from a timeslot, it is also unassigned from its assigned room (if any).

7.3.2.1 RemoveRandom

In this method, unassign time-moves are performed at random among all EventChains assigned a timeslot. This method will diversify the search.

7.3.2.2 RemoveRelated

This method is related to Shaw operator (Shaw (1997, 1998)). The basic idea is to remove a set of events which are *related* according to some measure. Compared to a set of random events, related events have a higher probability for the possibility of swapping timeslots, which can potentially lead to better solutions. Let $\mathcal{A}(ec)$ and $\mathcal{C}(ec)$ denote the set of entities and classes of EventChain ec, respectively. $R_{\mathcal{A}} \in [0,1]$ and $R_{\mathcal{C}} \in [0,1]$ denotes scaling parameters which require tuning. The amount of randomness in the selection is determined by $p_{\text{related}} \in [1,\infty]$. The related measurement is in this case defined as the percentage overlap among entities, and classes between two EventChains, defined by eq. (7.49) for EventChains ec and ec',

$$\mathfrak{R}_{e,e'} = R_{\mathcal{A}} \frac{|\mathcal{A}(ec) \cup \mathcal{A}(ec')|}{\min\left(|\mathcal{A}(ec)|, |\mathcal{A}(ec')|\right)} + R_{\mathcal{C}} \frac{|\mathcal{C}(ec) \cup \mathcal{C}(ec')|}{\min\left(|\mathcal{C}(ec)|, |\mathcal{C}(ec')|\right)} \tag{7.49}$$

Algorithm 2 shows the RemoveRelated procedure. Let $E(M)$ denote the number of events of the EventChain of move M.

Algorithm 2 RemoveRelated

```
input: A feasible solution S, and remove-quantity q
q̄ = 0
ec = a random selected chain assigned to a timeslot
D_done = {ec}
while q̄ < q do
    ec' = randomly selected from D_done
    L = all EventChains assigned to a timeslot, sorted by similarity to ec'
    choose a random number y ∈ [0; 1[
    ec = element number y^{p_related} |L| of L
    apply M^{time}_{ec,t} to S, where t is the timeslot assigned to EventChain ec
    D_done = D_done ∪ ec
    q̄ = q̄ + E(M)
end while
return q̄
```

7.3.2.3 RemoveTime

This remove method removes events assigned to the same timeslot, by the following procedure: First select some random timeslot. Now remove EventChains assigned to this timeslot, until q EventChains have been removed. If at some point no more EventChains are assigned to the timeslot, select a new random timeslot.

7.3.2.4 RemoveClass

Select a random class, and remove EventChains which contain it from their respective timeslot, until q EventChains have been removed. If at some point no more EventChains are assigned to this class, select a new random class.

7.3.3 Coupled destroy/repairs

Coupling certain destroy methods with certain repair methods is a small extension of the ALNS framework. This implies that the logic for choosing certain destroy/repair methods are extended, such that also certain pairs of methods can be chosen. This is useful for specialized destroy/repair methods, where a specific part of the solution is destroyed, and a competitive solution is not expected unless this part of the solution is repaired. In the following we describe a neighborhood where coupling seems useful, namely InsertRoom and RemoveRoom.

7.3.3.1 InsertRoom

In InsertRoom, rooms are assigned to events in a greedy way. This means that the best assign-room move $M_{e,r}^{\text{room}}$is found for any event $e \in \mathcal{E}$ and room $r \in \mathcal{R}$. If this move results in a better solution, i.e. if $\Delta\left(M_{e,r}^{\text{room}}\right) < 0$, apply it to the solution, and otherwise stop.

7.3.3.2 RemoveRoom

In RoomRemove, q random room-assignments are removed from the solution, i.e. q random unassign-room moves are performed.

7.3.4 Parameter Tuning

A basic implementation of F-Race (Birattari (2005); Balaprakash et al. (2007)) is used to tune parameters for best algorithmic performance. Given a set of tuning-instances (different from those datasets used to establish computational results), this algorithm is capable of automatically tuning optimization algorithms, i.e. finding the optimal set of values for all free parameters in the algorithm. In our experiments, the values obtained in Kristiansen et al. (2013) were used as a starting point, and we allowed F-Race to run for 24 hours. This timelimit was not sufficient for finding the optimal set of weights, and we report here the best found parameter-configuration. Table 7.1 shows the chosen value for each parameter.

Parameter	w_{SA}	d_{SA}	N_{it}	ρ	σ	$R_{\mathcal{A}}$	$R_{\mathcal{C}}$	p_{related}	$p_{\text{start}}^{\text{des}}$	$p_{\text{start}}^{\text{des}}$
Domain	$]0;1[$	$]0;1[$	$[0;\infty]$	$[0;1]$	$[0;\infty]$	$[0;1]$	$[0;1]$	$[1;\infty]$	$[0;1]$	$[0;1]$
Value	0.01	0.99	100	0.3	10000	0.7	0.3	20	0.10	0.01

Table 7.1: List of parameters and their tuned value

7.4 Results

The purpose of this section is to compare and evaluate the described solution approaches. A variety of datasets are therefore selected from the database of the commercial product Lectio. Currently, this database contains almost 5000 potential datasets from 110 different high schools. We select 100 of these randomly for evaluating the algorithms, see (Sørensen and Stidsen, 2013, Table 2 p. 26) for more details and statistics. Using these datasets, we aim at answering these question:

- How does the found solutions compare with the bounds obtained from the IP-based solution approaches? This is a way of evaluating the quality of the obtained solutions, and/or the bounds obtained by the MIP approaches.
- Which solution approach obtains best solutions within a short timeframe? This is important in practice for the high schools, as they expect an algorithm capable of producing good timetables quickly.

In all cases Gurobi 5.01 has been used as MIP-solver, and tests were run in C# 4.5 using nUnit 2.6 on Windows 8 64bit. The machine was equipped with an Intel i7 CPU clocked at 2.80GHz and with 12GB of RAM. As initial solution ('MIPStart' parameter), events were assigning to either their locked timeslot/room or the dummy-timeslot/room. The percentage-gap between an objective value z and a lower bound LB is calculated by $\text{gap} = 100\frac{z-LB}{z}$.

7.4.1 Solution approach comparison

In this section a comparison between the proposed solution approaches is performed with the following goals: 1) Compare the solutions obtained by each solution approach given a high time-limit. This is important for evaluating the potential of each approach. 2) Evaluate the solution approaches in terms of the bounds obtained by solving the MIP and the two-stage MIP. This is an important measure of solution quality.

The default parameter settings of Gurobi were used. The maximum number of threads and time-limits were set as follows:

	MIP	2-stage MIP	ALNS
Max. CPU threads	8	8	1
Time limit (s)	7200	6480 / 720	240
No. of runs	1	1	10

As we are interested in obtaining good bounds from the MIP approaches, these are allowed more computational time and more CPU threads. This means the comparison of solution quality favors the MIP approaches. In the next section, a more direct comparison of solution quality is made. It should be noted that the described version of ALNS has no parallelization implemented, so it would not benefit from more threads. A parallel version is considered for future work (see e.g. Ropke (2009)).

Table 7.2 shows the obtained results (a summary is given in the last rows in the table). This shows that the ALNS heuristic finds the best solution in 78 cases. In no cases are the pure MIP approach best, and in 20 cases are the two-stage approach best. In those cases where the two-stage approach is best, the dataset is usually small in terms of number of events.

To establish statistical significance of the results, the non-parametric statistical test *Wilcoxon Signed-Rank* has been used. Given a number of pairs of values, the null-hypothesis of the test is that the median difference between the pairs is zero. For the values in Table 7.2, this gives the

following results. The null-hypothesis when comparing MIP/Two-stage MIP, MIP/ALNS and Two-Stage MIP/ALNS can in all cases be rejected with confidence $> 99\%$. This means that the performance differences are statistical significant.

It was expected that the two-stage approach would outperform the pure MIP, but it is surprising that the ALNS heuristic outperforms both MIP approaches. On the larger datasets, the ALNS algorithm generally performs best. We expect this is due to the fact that ALNS is not prone to the *curse of dimensionality*, and overall scales better with the size of the datasets. The exact reasoning for this behavior is hard to pin-point, but it is a common pattern when comparing heuristics with exact methods.

In total, a lower bound was found for 79 datasets. The MIP was able to find a lower bound in 46 cases, whereas the two-stage approach found a bound in 79 cases. This means that for the two-stage model, Gurobi was not able to solve the LP-relaxation of the root node in the stage one model in 21 cases, which is surprising. The model is not numerically instable, so currently our best guess is that we are facing issues with degeneracy. In 33 cases, the bound obtained by the MIP were best, and in 46 cases the bound obtained by the two-stage model were best. Note that if the root LP-relaxation was not solved for a specific dataset, the reported solution is equal to the initial solution. In case the root LP-relaxation is not solved, the ALNS algorithm clearly performs better.

For those instances where a bound is found, the gap obtained for the ALNS is in average 25.6%, which seems rather high. Especially considering that those instances where a gap is not found are the big instances, which we expect are more difficult to solve. The inevitable question arises whether this is due to a poor bound, or due to poor solution quality. Future research will hopefully shed light upon this matter.

Figure 7.5a shows the linear regression of objectives as a function of number of events in dataset. This shows that as the size of datasets grows, the performance advantage of ALNS compared to the other solution approaches increases. This means that the ALNS heuristic scales better with the size of the datasets. Figure 7.5b summarizes key measurements from Table 7.2.

Table 7.2: Comparison of solution approaches. For the MIP model, column 'Time' shows the runtime, column 'Obj' shows the objective of the obtained solution, column 'LB' shows the lower bound, and 'Gap' shows the gap between the objective and the bound found by the MIP and the two-stage MIP. For the two-stage MIP, column 'Stg1' and 'Stg2' shows the elapsed time for solving the first and second stage, respectively. The remaining columns are analogous to the those defined for the MIP approach. For ALNS is shown the average objective found over 10 runs 'Obj', the standard deviation of these runs 'σ', and 'Gap' denotes the gap between the average objective and the best bound found. When a bound or solution is not found within the timelimit, a dash is written. The best solution for each dataset is written in **bold font** (skipping draws).

	MIP				Two-stage MIP					ALNS		
Dataset	Time	Obj	LB	Gap	Stg1	Stg2	Obj	LB	Gap	Obj	σ	Gap
AalborTG2012	>7200	6118	5946	2.8	>6480	1	**6018**	5934	1.2	6317	66.6	5.9
AarhusA2011	>7200	58015	-	89.7	>6480	93	15872	5986	62.3	**10037**	387.2	40.4
AarhusA2012	>7200	17096	5722	64.9	>6480	>720	8947	6005	32.9	**7971**	87.9	24.7
Aars2009	>7200	49504	-	76	>6480	6	20780	11874	42.9	**14900**	154	20.3
Aars2010	>7200	81970	-	84	>6480	15	25057	13134	47.6	**16268**	158.8	19.3
Aars2011	>7200	77967	-	87.6	>6480	11	30623	9709	68.3	**14256**	287.1	31.9
Aars2012	>7200	55049	-	86.5	>6480	4	21206	7456	64.8	**10701**	99.1	30.3
Alssund2010	>7200	52717	-	87.1	>6480	27	23173	6811	70.6	**9967**	438.5	31.7
Alssund2012	>7200	108810	-	-	>6480	2	108810	-	-	**29803**	609.9	-
BagsvaG2010	>7200	6777	3171	53.2	>6480	14	**3916**	3063	19	3960	87.8	19.9

Continued on next page

Table 7.2 – continued from previous page

	MIP				Two-stage MIP					ALNS		
Dataset	Time	Obj	LB	Gap	Stg1	Stg2	Obj	LB	Gap	Obj	σ	Gap
BirkerG2011	>7200	119600	-	-	>6480	1	119600	-	-	**42063**	751.9	-
BirkerG2012	>7200	110180	-	85.8	>6480	>720	**19322**	15662	18.9	19552	54.1	19.9
BjerrG2009	>7200	52639	-	78.9	>6480	8	35514	11094	68.8	**16877**	271	34.3
BjerrG2010	>7200	12868	3928	69.5	>6480	22	5788	3868	32.1	**4983**	74	21.2
BjerrG2011	>7200	13009	4142	68.2	>6480	>720	9302	4060	55.5	**6334**	119.1	34.6
BjerrG2012	>7200	17200	5055	70.6	>6480	354	15265	5007	66.9	**8023**	220.3	37
BroendG2012	>7200	2005	1881	6.2	1173	14	**1929**	1859	2.5	2040	30	7.8
CPHWGym2010	>7200	34415	-	89.1	>6480	2	19363	3759	80.6	**6775**	328.6	44.5
CPHWGym2011	>7200	38232	-	89.3	>6480	2	16212	4095	74.7	**5679**	179.3	27.9
CPHWGym2012	>7200	40945	-	89.7	>6480	2	15543	4205	73	**6762**	217.7	37.8
CPHWHG2012	>7200	46625	8157	82.1	>6480	10	23088	8338	63.9	**11077**	227.6	24.7
CPHWHTX2010	>7200	27174	9179	66.2	>6480	10	15943	8828	42.4	**11342**	146.4	19.1
CPHWHTX2011	>7200	22466	20460	8.9	>6480	5	**20708**	18490	1.2	20734	27.6	1.3
CPHWHTX2012	>7200	25998	14481	44.3	>6480	13	21392	13115	32.3	**16256**	126.1	10.9
DetFG2012	>7200	8017	7168	10.6	>6480	6	**7265**	7018	1.3	7560	73.4	5.2
DetKG2010	>7200	6058	1732	69.9	>6480	1	4006	1821	54.5	**2947**	69	38.2
DetKG2011	>7200	5594	1732	68.2	>6480	14	4366	1780	59.2	**2820**	136	36.9
EUCN2009	>7200	7557	2911	61.5	>6480	2	4298	2856	32.3	**3737**	116	22.1
EUCN2010	>7200	4231	3329	21.3	>6480	32	**3463**	3246	3.9	3882	76	14.2
EUCN2011	>7200	1435	1395	2.8	>6480	1	**1430**	1384	2.5	1468	13	4.9
EUCN2012	>7200	9430	2327	74.9	>6480	1	5059	2363	53.3	**3289**	160	28.2
EUCNHG2010	>7200	1476	1371	7.1	>6480	5	**1421**	1368	3.5	1505	28	8.9
EUCS2012	>7200	4689	3576	23.7	>6480	12	3783	3347	5.5	**3714**	32	3.7
FaaborgG2008	>7200	125330	-	-	>6480	54	125330	-	-	**68124**	2156	-
FalkonG2009	>7200	88890	-	-	>6480	0	88890	-	-	**10449**	251	-
FalkonG2011	>7200	76170	-	93.2	>6480	>720	16543	5183	68.7	**8584**	271	39.6
FalkonG2012	>7200	100190	-	93.9	>6480	>720	16666	6105	63.4	**10143**	432	39.8
GUAasia2010	>7200	6579	6354	3.4	6	>720	**6461**	6035	1.7	6527	7	2.7
GUQaqor2011	>7200	19623	4537	76.8	>6480	6	10005	4554	54.5	**6674**	301	31.8
GUQaqor2012	>7200	11488	4314	62.5	>6480	15	7619	4294	43.4	**5733**	134	24.8
HadersK2011	>7200	51190	-	92.4	>6480	>720	14229	3909	72.5	**7128**	386	45.2
HasserG2010	>7200	96790	-	-	>6480	0	96790	-	-	**11963**	132	-
HasserG2011	>7200	99840	-	-	>6480	1	99840	-	-	**16061**	472	-
HasserG2012	>7200	112160	-	-	>6480	2	112034	-	-	**18338**	672	-
HerningG2010	2	37	37	0	0	2	37	35	0	37	0	0
HerningG2011	>7200	163785	-	94	>6480	12	23117	9829	57.5	**15091**	144	34.9
HerningG2012	>7200	185433	-	94.7	>6480	>720	14952	9763	34.7	**13147**	76	25.7
HoejeTaG2008	>7200	6292	2253	59.3	>6480	1	**2707**	2563	5.3	2958	92	13.4
HoejeTaG2009	>7200	45260	-	87.2	>6480	470	26066	5773	77.9	**9157**	303	37
HoejeTaG2010	>7200	45095	-	86.3	>6480	116	25678	6188	75.9	**9862**	232	37.3
HoejeTaG2011	>7200	51050	-	86.8	>6480	48	32630	6726	79.4	**10158**	201.5	33.8
HoejeTaG2012	>7200	72455	7592	89.2	>6480	224	18627	7845	57.9	**12502**	143.4	37.3
HorsenS2009	63	3100	3100	0	0	2	3100	2865	0	3111	9	0.4
HorsenS2012	>7200	86090	-	-	>6480	0	86090	-	-	**10056**	434.9	-
Johann2012	>7200	92575	-	80.1	>6480	>720	27781	18456	33.6	**23001**	193.4	19.8
KalundG2011	>7200	126150	-	-	>6480	1	126150	-	-	**38479**	514.5	-
KalundG2012	>7200	123010	-	-	>6480	1	123010	-	-	**26768**	348.8	-
KalundHG2010	>7200	12103	4540	62.4	>6480	16	6351	4551	28.3	**5631**	83.1	19.2
KoebenPG2012	>7200	1872	637	65.7	>6480	2	**874**	642	26.5	888	31	27.7
KoegeH2012	>7200	108347	-	91.6	>6480	2	20150	9096	54.9	**11418**	136.2	20.3
KongshoG2010	>7200	8889	2411	72	>6480	3	7954	2488	68.7	**4296**	175.8	42.1
MariageG2009	>7200	54030	-	90.5	>6480	161	20138	5118	74.6	**8013**	251.6	36.1
MorsoeG2012	>7200	42762	-	91	>6480	54	10241	3854	62.4	**5651**	122.6	31.8
NaerumG2008	>7200	118370	-	-	>6480	0	117894	-	-	**24104**	502.8	-
NaerumG2009	>7200	100450	-	94.9	209	>720	**6681**	5114	23.5	7667	62.5	33.3
NielsSG2011	>7200	10464	3323	67.4	>6480	2	6132	3412	44.4	**4953**	111.7	31.1
NielsSG2012	>7200	12747	5722	55	>6480	16	8003	5738	28.3	**6952**	107.4	17.5

Continued on next page

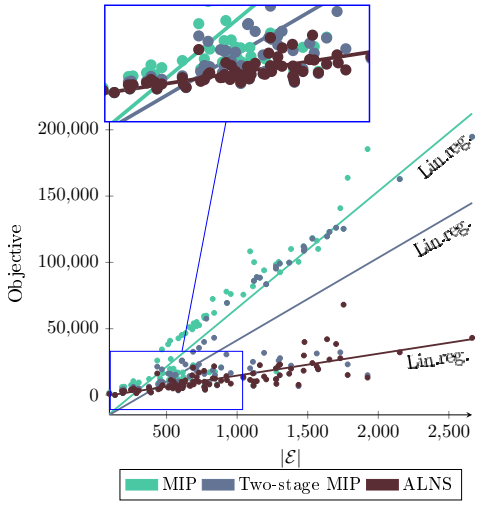
Table 7.2 – continued from previous page

	MIP				Two-stage MIP					ALNS		
Dataset	Time	Obj	LB	Gap	Stg1	Stg2	Obj	LB	Gap	Obj	σ	Gap
NordfynG2012	>7200	8201	4152	49.4	>6480	205	**4890**	4048	15.1	5160	38.6	19.5
NyborgG2011	>7200	94059	-	93.5	>6480	>720	31809	6129	80.7	**13944**	434.4	56.1
OdderCfU2010	>7200	59540	-	79.5	>6480	1	40032	12188	69.6	**18219**	189.2	33.1
OdderG2009	>7200	59851	-	-	>6480	2	57586	-	-	**9308**	206.8	-
OdderG2012	>7200	17402	9602	44.8	>6480	>720	14888	8878	35.5	**12307**	157.8	22
OrdrupG2010	>7200	75700	-	85.9	>6480	37	**12936**	10665	17.6	13663	391.8	21.9
OrdrupG2011	>7200	116400	-	85.5	>6480	>720	31329	16904	46	**21612**	630.4	21.8
RibeK2011	>7200	61945	-	73.8	>6480	390	43175	16209	62.5	**21679**	260.1	25.2
RysenG2010	>7200	110690	-	-	>6480	1	110690	-	-	**39971**	148	-
RysenG2011	>7200	100313	-	82.3	>6480	>720	25989	17756	31.7	**22260**	99.1	20.2
RysenG2012	>7200	110111	-	86.3	>6480	>720	22156	15115	31.8	**19841**	189.2	23.8
SanktAG2012	>7200	4624	3415	26.2	75	>720	**3911**	3376	12.7	4207	33.5	18.8
SkanderG2010	>7200	7708	6051	21.5	77	>720	**6875**	5712	12	7209	36.5	16.1
SkanderG2011	>7200	88470	-	-	>6480	0	88470	-	-	**22525**	368.5	-
SkanderG2012	>7200	98487	-	-	>6480	1	95319	-	-	**20138**	682.8	-
SkiveG2010	>7200	194740	-	-	>6480	21	194740	-	-	**43120**	1261.2	-
SlagelG2012	>7200	162960	-	-	>6480	36	162765	-	-	**32167**	1225.7	-
SoendS2011	>7200	83560	-	-	>6480	1	83560	-	-	**11776**	248.4	-
SoendS2012	>7200	17778	6838	61.5	>6480	2	11915	6647	42.6	**8420**	94	18.8
StruerS2012	>7200	-	-	-	>6480	18	207488	-	-	**73361**	3188	-
VardeG2012	>7200	20933	5921	71.7	>6480	13	20622	5720	71.3	**10777**	1911	45.1
VejenG2009	>7200	69450	-	-	>6480	0	69450	-	-	**11264**	209	-
Vejlefjo2011	>7200	52035	-	83.6	>6480	>720	18043	8511	52.8	**13514**	183	37
VestfynG2009	>7200	11606	4176	64	>6480	347	5999	4137	30.4	**5973**	148.5	30.1
VestfynG2010	>7200	16895	4308	74.5	>6480	354	**5974**	4225	27.9	6761	211.3	36.3
VestfynG2011	>7200	13624	5110	62.5	>6480	25	**6657**	4925	23.2	7013	218.7	27.1
VestfynG2012	>7200	11095	4279	61.4	>6480	48	**5212**	4210	17.9	5244	52.5	18.4
ViborgK2011	>7200	99170	-	-	>6480	0	99170	-	-	**14923**	406.1	-
ViborgTG2009	>7200	19891	8695	56.3	>6480	33	12077	8356	28	**10216**	102	14.9
ViborgTG2010	>7200	12727	4130	67.6	>6480	112	10226	3990	59.6	**4932**	66	16.3
ViborgTG2011	>7200	16433	6716	59.1	>6480	46	9808	6204	31.5	**7478**	40	10.2
VirumG2012	>7200	140883	-	87.4	>6480	>720	32183	17770	44.8	**27738**	502	35.9
VordingbG2009	>7200	17025	5457	68	>6480	91	9905	5243	44.9	**8568**	97	36.3
Avg.†				65.3					41.3			25.6
Max.				94.9					80.7			56.1
No. times best		0					20			78		
No. bound found			46					79				
No. best bound			33					46				

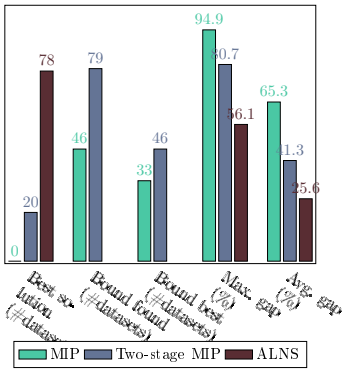
† Rows where either of the gap columns are not available are skipped for a fair comparison.

7.4.2 Operational considerations

In this section it is considered what approach can be used to find the best solutions within a short time horizon. This means that for the MIP approaches, we are not interested in the bound, but only in actual solutions. According to the documentation of Gurobi, some parameters will make the solver focus on finding good solutions. Table 7.3 shows the chosen values for these parameters, which have been selected ad-hoc on the basis of the documentation of Gurobi and with the help of Gurobi Support. Furthermore the maximum number of threads for Gurobi is set to 1, to obtain a fair comparison with the ALNS heuristic. In our experience, the high schools expect that an algorithm should produce a timetable within a fairly small time horizon. Table 7.4 shows for each dataset the best found solution by each solution approach after 240, 420 and 600 seconds. The table shows that the ALNS algorithm finds better solutions for all three time



(a) Objective of datasets as a function of number of events. Linear regression for each set of points is also shown.



(b) Bar plot summary from Table 7.2.

Figure 7.5: Performance of the three solution approaches illustrated.

Table 7.3: Gurobi parameter settings in operational setting

	Default	Value	Intention
MIPFocus	0	1	Focus on finding good solutions
Heuristics	0.05	0.90	Attempt to spend 90% of solver time on heuristics
ImproveStartTime	infinity	0	Immediately focus on solution quality
Cuts	-1	0	Turn off all cuts

horizons in the majority of cases (88, 86 and 87, respectively). Furthermore ALNS is able to find a feasible solution in all cases, and is the best algorithm in an operational setting.

Table 7.4: Best found solutions for the three solution approaches at certain points in time.

	240s			420s			600s		
Dataset	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS
AalborTG2012	8530	**6182**	6296	8530	**6069**	6278	7776	**6067**	6259
AarhusA2011	58015	58015	**10244**	58015	58015	**10063**	58015	58015	**9995**
AarhusA2012	66350	12121	**8013**	66350	10966	**7907**	66350	10738	**7880**
Aars2009	-	49484	**15068**	49504	49484	**14900**	49504	49484	**14835**
Aars2010	-	81970	**16474**	-	30344	**16250**	-	30344	**16127**
Aars2011	-	64774	**14511**	77967	64774	**14230**	77967	64774	**14150**
Aars2012	-	52520	**10674**	55049	25984	**10544**	55049	25252	**10510**
Alssund2010	52717	52585	**9825**	52717	52585	**9714**	52717	52585	**9658**
Alssund2012	-	108810	**30349**	-	108810	**29088**	-	108810	**28530**
BagsvaG2010	9212	5855	**3942**	9212	5400	**3939**	8142	5247	**3939**
BirkerG2011	-	119600	**42190**	-	119600	**41415**	119600	119600	**41066**
BirkerG2012	-	**19526**	19642	-	**19497**	19566	-	19464	**19434**
BjerrG2009	52639	52395	**17152**	52639	52395	**16846**	52639	52395	**16716**
BjerrG2010	44901	7327	**4923**	14512	7118	**4889**	14512	7052	**4885**
BjerrG2011	52933	11371	**6339**	52933	11059	**6302**	52933	10796	**6257**
BjerrG2012	39745	39745	**7974**	39745	39745	**7838**	39745	39745	**7797**
BroendG2012	2263	**1935**	2040	2207	**1935**	2036	2002	**1935**	2036

Continued on next page

Table 7.4 – continued from previous page

	240s			420s			600s		
Dataset	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS
CPHWGym2010	34415	34415	**6670**	34415	34415	**6583**	34415	34415	**6560**
CPHWGym2011	38232	37368	**5713**	38232	16573	**5628**	38232	16573	**5601**
CPHWGym2012	40945	37095	**6554**	40945	37095	**6508**	40945	21405	**6487**
CPHWHG2012	46625	46099	**11023**	46625	46099	**10954**	46625	46099	**10919**
CPHWHTX2010	38497	21441	**11293**	38497	21441	**11215**	38497	21441	**11199**
CPHWHTX2011	30578	21508	**20745**	24403	21349	**20729**	24403	21024	**20724**
CPHWHTX2012	31860	27169	**16137**	31860	26028	**16096**	31860	25441	**16090**
DetFG2012	19736	**7460**	7583	14147	**7417**	7557	13618	**7415**	7546
DetKG2010	7556	6139	**2951**	7556	6046	**2949**	7418	6041	**2949**
DetKG2011	14445	6360	**2863**	6653	6348	**2848**	6653	6185	**2846**
EUCN2009	18590	5833	**3680**	18590	5693	**3672**	7856	5335	**3671**
EUCN2010	5958	3961	**3940**	5947	**3873**	3916	5947	**3873**	3910
EUCN2011	**1451**	1484	1479	**1448**	1478	1479	**1448**	1478	1479
EUCN2012	22570	8262	**3253**	22570	7549	**3227**	22570	6993	**3223**
EUCNHG2010	1847	**1443**	1488	1842	**1442**	1484	1842	**1442**	1484
EUCS2012	6629	4229	**3710**	5169	4078	**3707**	5169	4078	**3706**
FaaborgG2008	-	125330	**69675**	-	125330	**64777**	-	125330	**62082**
FalkonG2009	-	88890	**10541**	-	88890	**10212**	-	88890	**10098**
FalkonG2011	-	76170	**8629**	76170	76170	**8375**	76170	76170	**8286**
FalkonG2012	-	82629	**10153**	100190	82629	**9786**	100190	82629	**9688**
GUAasia2010	-	**6466**	6534	38850	**6462**	6515	38850	**6457**	6509
GUQaqor2011	38210	11910	**6749**	38210	11795	**6625**	38210	11810	**6577**
GUQaqor2012	42346	10810	**5780**	42346	9306	**5694**	42346	9296	**5666**
HadersK2011	51190	51190	**7156**	51190	51190	**7029**	51190	51190	**6974**
HasserG2010	-	96790	**12061**	96790	96790	**11645**	96790	96790	**11491**
HasserG2011	-	99840	**15970**	-	99840	**15479**	99840	99840	**15280**
HasserG2012	-	112034	**19099**	-	112034	**18379**	-	112034	**18016**
HerningG2010	-	**37**	37	-	**37**	37	-	**37**	37
HerningG2011	163785	102119	**15971**	163785	26648	**15602**	163785	26646	**15347**
HerningG2012	185433	-	**13290**	185433	28140	**13117**	185433	28140	**12964**
HoejeTaG2008	14865	4950	**2941**	6923	4646	**2921**	6923	4175	**2920**
HoejeTaG2009	-	45260	**9275**	45260	45260	**9118**	45260	28064	**9079**
HoejeTaG2010	-	45095	**9733**	45095	32686	**9644**	45095	28120	**9615**
HoejeTaG2011	-	51050	**10168**	-	51050	**10065**	51050	51050	**10021**
HoejeTaG2012	-	30074	**12468**	-	30074	**12349**	-	25206	**12298**
HorsenS2009	3100	**3100**	3115	3100	**3100**	3115	3100	**3100**	3115
HorsenS2012	-	86090	**10181**	-	86090	**9733**	86090	86090	**9550**
Johann2012	-	92575	**23104**	-	31672	**22892**	92575	31507	**22780**
KalundG2011	-	126150	**39102**	-	126150	**38464**	126150	126150	**37929**
KalundG2012	-	123010	**27503**	-	123010	**26451**	-	123010	**25828**
KalundHG2010	27530	7981	**5631**	12214	7879	**5599**	12008	7758	**5596**
KoebenPG2012	2517	1589	**876**	2517	1483	**876**	2517	1333	**876**
KoegeH2012	-	26279	**11431**	108347	22745	**11338**	108347	21005	**11274**
KongshoG2010	34265	10249	**4302**	34265	8675	**4278**	34265	8602	**4268**
MariageG2009	54030	54030	**8152**	54030	54030	**8045**	54030	54030	**7962**
MorsoeG2012	42762	42395	**5681**	42762	42395	**5627**	42762	42395	**5609**
NaerumG2008	-	117894	**24543**	118370	117894	**23776**	118370	117894	**23311**
NaerumG2009	-	**6752**	7767	-	**6752**	7679	-	**6746**	7545
NielsSG2011	-	10828	**4852**	19200	10486	**4799**	19200	8954	**4796**
NielsSG2012	50253	11041	**6945**	50253	10756	**6902**	50253	10752	**6848**
NordfynG2012	63210	6216	**5218**	63210	5909	**5158**	8227	5909	**5137**
NyborgG2011	-	85372	**14041**	-	85372	**13647**	94059	85372	**13430**
OdderCfU2010	-	59473	**18229**	-	59473	**18059**	-	59473	**17997**
OdderG2009	59851	57586	**9271**	59851	57586	**9037**	59851	57586	**8939**
OdderG2012	86520	16963	**12482**	86520	14977	**12282**	86520	14976	**12210**
OrdrupG2010	75700	75700	**13645**	75700	14806	**13465**	75700	13944	**13337**
OrdrupG2011	-	116400	**21986**	-	116400	**21798**	-	116400	**21541**

Continued on next page

Table 7.4 – continued from previous page

	240s			420s			600s		
Dataset	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS	MIP	2SMIP	ALNS
RibeK2011	-	61945	**21762**	61945	61945	**21544**	61945	61945	**21490**
RysenG2010	-	110690	**40194**	-	110690	**40010**	110690	110690	**39778**
RysenG2011	-	89741	**22391**	100313	89741	**22191**	100313	89741	**22006**
RysenG2012	-	95590	**20242**	-	95590	**19910**	110111	95590	**19598**
SanktAG2012	54080	**3824**	4252	54080	**3821**	4200	54080	**3819**	4172
SkanderG2010	-	**6893**	7320	-	**6878**	7234	-	**6876**	7152
SkanderG2011	-	88470	**22985**	-	88470	**22374**	-	88470	**22087**
SkanderG2012	-	95319	**20367**	-	95319	**19659**	-	95319	**19334**
SkiveG2010	-	194740	**43699**	-	194740	**42967**	-	194740	**42127**
SlagelG2012	-	162765	**30743**	-	162765	**30186**	-	162765	**29673**
SoendS2011	83560	83560	**12049**	83560	83560	**11674**	83560	83560	**11519**
SoendS2012	87883	14589	**8451**	16717	13920	**8355**	16717	12765	**8298**
StruerS2012	-	207488	**69927**	-	207488	**68367**	-	207488	**67294**
VardeG2012	60980	60980	**9684**	60980	60980	**9526**	60980	60980	**9455**
VejenG2009	-	69450	**11224**	-	69450	**10886**	69450	69450	**10779**
Vejlefjo2011	-	52035	**13478**	-	52035	**13338**	52035	52035	**13293**
VestfynG2009	62063	7914	**6117**	62063	**5867**	6037	62063	**5755**	6011
VestfynG2010	61216	11853	**6647**	61216	10155	**6562**	61216	9303	**6548**
VestfynG2011	67790	8577	**7068**	67790	8521	**6997**	67790	8269	**6936**
VestfynG2012	66096	7607	**5241**	66096	7354	**5187**	66096	7352	**5182**
ViborgK2011	-	99170	**15459**	-	99170	**14670**	-	99170	**14360**
ViborgTG2009	39385	21077	**10229**	39385	14309	**10156**	39385	13630	**10141**
ViborgTG2010	34980	12299	**4972**	34980	11733	**4943**	34980	10977	**4934**
ViborgTG2011	36300	11887	**7496**	36300	9723	**7474**	36300	9723	**7469**
VirumG2012	-	111119	**23561**	140883	111119	**23359**	140883	34158	**23176**
VordingbG2009	55115	11646	**8607**	55115	10953	**8535**	55115	10939	**8523**
No. solutions	55	99	100	70	100	100	81	100	100
No. times best	1	11	88	1	13	86	1	12	87

7.4.3 XHSTT Datasets

To make our datasets public available, a conversion scheme to the XHSTT format has been developed (see Sørensen and Stidsen (2013)). Most constraints of HSTTP are modeled analogously by XHSTT, but some important are not supported: Constraints (7.3′), (7.4′) and (7.24) only considers a subset of events w.r.t. entity conflicts, room conflicts and day conflicts for classes, respectively. In the current version of XHSTT such constraints consider the full set of events, which might lead to inevitable violations of hard constraints. Constraint (7.8) specifies that an event cannot be assigned a room unless it is assigned a timeslot, which is not possible in XHSTT. Constraint (7.26) excludes some classes from be checked from neighbor-day conflicts on certain days, which is also not possible in XHSTT (all neighbor-days will be penalized).

Currently, copyright issues have been settled with three schools, such that three datasets have been made available in the archive XHSTT-2013. We hope to be able to make more datasets publicly available soon. Table 7.5 shows statistics for the datasets converted into the XHSTT format. The heuristic described in Sørensen et al. (2012) is applied to all instances 10 times, each with a timelimit of 240 seconds, and the best solution found is shown in the table. It is seen that all found solutions contain hard constraint violations. As previously described, it is expected that in the majority of cases, the optimal solution will contain some violation of hard constraints.

Table 7.5: XHSTT datasets statistics.

Dataset	Times	Teach.	Rooms	Classes	Stud.	Events	Duration	Best sol.
FalkonG2012	50	91	63	313	278	1120	1120	(101,19464)
HasserG2012	50	100	69	423	521	1475	1475	(319,24312)
VejenG2009	60	46	53	189	163	928	928	(2,23275)

7.5 Conclusion

A complex Integer Programming model of timetabling for high schools in Denmark has been described. The model contains all constraints required in practice by a large number of high schools, and it is used by many schools to produce annual timetables. The model is $\mathcal{NP}$-hard, but a simple decomposition has been suggested which facilitates solution times. Furthermore, a heuristic based on Adaptive Large Neighborhood Search has been discussed, which yields a total of three different solution approaches.

Using 100 real-world datasets, these solution approaches have been evaluated in terms of lower bounds derived from the MIP approaches, and solution quality in a production setting. The ALNS heuristic proved to perform best wrt. both aspects.

The gap between the solutions found by ALNS and the best bound found is in average 25.6%, which is unsatisfactory. Future research will hopefully be able to narrow this gap, either by finding better bounds, or by strengthening the solution approaches.

The chosen problem formulation might leave events which are not assigned to a timeslot. This can happen either because the problem is too constrained due to the parameters set by the high school, or it can happen as a sub-optimal solution was provided by the algorithm. A way to tackle such unassigned events is a topic for future work. Currently we believe a solution could be to incorporate a different algorithm which is independent of the solution approaches described in this paper. This should be understood in the following sense. Once the algorithm to solve the HSTTP is finished, assume a number of events are left unassigned to timeslots. For one or several of these events, the high school can then attempt to find an alternative solution which assigns a timeslot to these events, such that the current solution does not change significantly. Such an algorithm could possibly be based on a concept like *Cyclic Transfers* (Post et al. (2012b)) or a variant of the *Repair Problem* for timetables (Kaneko et al. (1999); Kingston (2012)).

Bibliography

J. S. Appleby, D. V. Blake, and E. A. Newman. Techniques for producing school timetables on a computer and their application to other scheduling problems. *The Computer Journal*, 3(4): 237–245, 1961.

P. Avella and I. Vasil'Ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8:497–514, 2005. ISSN 1094-6136.

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

R. E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 19–24 2012.

A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70, April 2012. ISSN 0254-5330.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

P. D. Causmaecker and G. Berghe. Towards a reference model for timetabling and rostering. *Annals of Operations Research*, pages 1–10, 2010. ISSN 0254-5330.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

M. Gendreau and E. Burke, editors. *PATAT2008: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 18 - 22 August 2008.

C. C. Gotlieb. The construction of class-teacher timetables. In C. M. Popplewell, editor, *IFIP Congress*, volume 62, pages 73–77, North-Holland Pub. Co, 1962.

V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Technical Report CIRRELT-2011-42, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation, July 2011.

K. Kaneko, M. Yoshikawa, and Y. Nakakuki. Improving a heuristic repair method for large-scale school timetabling problems. In J. Jaffar, editor, *Principles and Practice of Constraint Programming*, volume 1713 of *Lecture Notes in Computer Science*, pages 275–288. Springer Berlin / Heidelberg, 1999. ISBN 978-3-540-66626-4.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

J. Kingston. The kts high school timetabling system. In E. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 308–323. Springer Berlin / Heidelberg, 2007.

J. H. Kingston. Repairing high school timetables with polymorphic ejection chains. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

D. Kjenstad, A. Riise, T. E. Nordlander, B. McCollum, and E. Burke, editors. *PATAT 2012: Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, 29 -31 August 2012.

P. Kostuch. The university course timetabling problem with a three-phase approach. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 109–125. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

N. L. Lawrie. An integer linear programming model of a school timetabling problem. *The Computer Journal*, 12(4):307–316, 1969.

M. Lundberg-Jensen, J. Bruun, and J. Ahmt. Task assignment for high school teachers. Technical report, Operations Management, Department of Informatics and Mathematical Modeling. Kgs. Lyngby. Technical University of Denmark, 2008.

S. Martello and P. Toth. An algorithm for the generalized assignment problem. *Operational research*, 81:589–603, 1981.

C. H. Martin. Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, 34(6):460–465, November 2004.

B. McCollum, E. Burke, and G. White, editors. *PATAT2010: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, 10 - 13 August 2010.

L. Muller. An adaptive large neighborhood search algorithm for the multi-mode resource-constrained project scheduling problem. l, Department of Management Engineering, Technical University of Denmark Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark, 2010.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3):230–238, 2003.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, S. Ahmadi, and F. Geertsema. Cyclic transfers in school timetabling. *OR Spectrum*, 34 (1):133–154, 2012b. ISSN 0171-6468.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012c.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke. Parallel large neighborhood search - a software framework. In *MIC 2009, The VIII Metaheuristics International Conference*, 2009.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

M. Salazar-Aguilar, A. Langevin, and G. Laporte. An adaptive large neighborhood search heuristic for a snow plowing problem with synchronized routes. In J. Pahl, T. Reiners, and S. Voss, editors, *Network Optimization*, volume 6701 of *Lecture Notes in Computer Science*, pages 406–411. Springer Berlin / Heidelberg, 2011. ISBN 978-3-642-21526-1.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

K. Schimmelpfeng and S. Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29:783–803, 2007. ISSN 0171-6468.

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

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

M. Sørensen and T. Stidsen. Comparing solution approaches for a complete model of high school timetabling. Technical Report 5.2013, DTU Management Engineering, Technical University of Denmark, March 2013.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

F. A. Tillman and T. M. Cain. An upperbound algorithm for the single and multiple terminal delivery problem. *Management Science*, 18(11):664 – 682, 1972. ISSN 00251909.

M. Yoshikawa, K. Kaneko, T. Yamanouchi, and M. Watanabe. A constraint-based high school scheduling system. *IEEE Expert*, 11(1):63 –72, feb 1996. ISSN 0885-9000.

E. Özcan. Towards an xml-based standard for timetabling problems: Ttml. In G. Kendall, E. K. Burke, S. Petrovic, and M. Gendreau, editors, *Multidisciplinary Scheduling: Theory and Applications*, pages 163–185. Springer US, 2005. ISBN 978-0-387-27744-8.

Chapter 8 Paper C

A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark

Matias Sørensen[1,2], Florian H. W. Dahms[3]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark
[3]Chair of Operations Research, RWTH Aachen University, Kackertstraße 7, 52072 Aachen,
Germany

Abstract Integer Programming (IP) has been used to model educational timetabling problems since the very early days of Operations Research. It is well recognized that these IP models in general are hard to solve, and this area of research is dominated by heuristic solution approaches. In this paper a *Two-Stage Decomposition* of an IP model for a practical case of high school timetabling is shown. This particular timetabling problem consists of assigning lectures to both a timeslot and a classroom, which is modeled using a very large amount of binary variables. The decomposition splits this model into two separate problems (Stage I and Stage II) with far less variables. These two separate problems are solved in sequence, such that the solution for the Stage I model is given as input to the Stage II model, implying that irreversible decisions are made in Stage I. However, the objective of the Stage II model is partly incorporated in the Stage I model by exploiting that Stage II can be seen as a *minimum weight maximum matching* problem in a bipartite graph. This theoretically strengthens the decomposition in terms of global optimality. The approach relies on Hall's theorem for the existence of matchings in bipartite graphs, which in its basic form yields an exponential amount of constraints in the Stage I model. However, it is shown that only a small subset of these constraints is needed, making the decomposition tractable in practice for IP solvers. To evaluate the decomposition, 100 real-life problem instances from the database of the high school ERP system Lectio are used. Computational results show that the decomposition performs significantly better than solving the original IP, in terms of both found solutions and bounds.

8.1 Introduction

Integer Programming (IP) has been used to model educational timetabling problems since the very early days of Operations Research (see e.g. Gotlieb (1962) and Lawrie (1969)). It is well recognized that these IP models in general are hard to solve (most forms of educational timetabling are in fact $\mathcal{NP}$-hard (Bardadym (1996))), and this area of research is dominated by heuristic solution approaches.

In this paper a large IP model for a real-world case of high school timetabling is considered, which has previously been shown to be a challenge for state-of-the-art MIP solvers. We consider a basic version of this IP, which includes the essential constraints of most timetabling problems. An innovative decomposition of this model is shown, which proves to be more efficient to solve.

When facing a hard IP model, decomposition is a commonly used tool to help speed up the solution procedure. Perhaps the most successful decomposition method in recent years is Column Generation (CG). However, not many papers on CG and timetabling models are found in the literature, and it seems that only relatively small instances have been attempted. Papoutsis et al. (2003) uses CG to solve a Greek case of high school timetabling, with the largest instance containing 9 class section, 21 teachers and 306 teaching hours. Santos et al. (2012) handle larger instances, but only generate lower bounds. Qualizza and Serafini (2005) describe a CG procedure for a university timetabling problem with 63 courses and 25 timeslots. The real-world instances considered in this paper are of much larger size.

A crucial part of a CG procedure is the identification of a block-diagonal structure in the problem, otherwise the CG procedure is most likely not efficient. For the high school timetabling problem described in this paper, it has not been possible to identify such a structure. Therefore this paper shows a different type of decomposition, a *Two-Stage Decomposition* (TSD). Such an approach was first used for timetabling applications in Lach and Lübbecke (2008) and Lach and Lübbecke (2012) with great success for the *curriculum-based university course* timetabling problem. The goal of this paper will be to modify the aforementioned approach to be applicable for the high school timetabling problem - giving special attention to the high school system in Denmark.

The considered timetabling problem essentially consists of assigning lectures to rooms and timeslots, which is commonly modeled using a very large amount of binary variables. There are three key points to the TSD:

- By substitution, the total amount of variables is significantly reduced, while linearity is maintained.
- Instead of solving the entire model at once, it can be solved in a two-stage fashion. I.e. both the set of variables and constraints are divided into two distinct sets, corresponding to two smaller IPs (denoted *Stage I* and *Stage II*, respectively).
- It will be evident that, except for two soft-constraints, this decomposition maintains optimality of the original model.

The outline of the TSD is to first solve Stage I, which provides a solution where lectures are assigned to timetslots. This partial solution is given as input to Stage II, which will assign rooms to the lectures, obtaining a solution for the original problem. The drawback of this decomposition is that the timeslots assigned to lectures in Stage I are considered as fixed by the Stage II model, which might prevent an optimal allocation of rooms to lectures. However, by exploiting the structure of the Stage II model, the Stage I model can be constrained in such a way that some penalties for assigning rooms to lectures are handled implicitly. Note that if all penalties for room assigning could be handled implicitly, the approach would be exact. However, two soft-constraints are not fully incorporated, so only a lower bound on the room penalties are known by the Stage I model. In fact, one of these soft-constraints are not handled at all by the described approach. Despite this, it seems likely that incorporating this lower bound in the Stage I model will provide better results overall (assuming that computing the lower bound does not have very bad influence on the computational efforts of the used IP solver). I.e. instead of the Stage I model being completely unaware of the penalties for room allocation, it seems better to at least incorporate some of them. Furthermore, the decomposition of the problem into two smaller

problems presents a big advantage in terms of reduction in the number of variables. Therefore the overall benefits of the TSD out-weight the downsides, and computational results will show that it is indeed way more effective than solving the original IP.

The contributions of this paper are the following: 1) It is shown that the approach from Lach and Lübbecke (2008) can also be applied to a high school timetabling problem originating from a practical setting, and by extensive computational results it is argued that the TSD is more effective than solving the original IP. Notice that a similar composition is briefly mentioned in Sørensen and Stidsen (2013) for the same high school timetabling problem, but this paper enhances the approach such that the theoretical maximum gap from optimality is narrowed. The presented approach turns out to be the most efficient exact algorithm for the problem so far. 2) Generally, it is shown how this type of decomposition can be applied to models with set-packing structure, by modifying the underlying equations originating from Hall's Theorem for matchings in bipartite graphs. 3) It is shown how the room-priorities of lectures can be handled, by adding a lower bound on the corresponding penalties to the Stage I model. This facilitates the quality of the solutions found, as shown by the computational results.

We expect that the basic structure required for applying the TSD can be found in other timetabling problems as well, and therefore the decomposition can potentially be used more broadly than the case of high school timetabling shown in this paper. This seems likely because the essential constraints used in the decomposition are among the most common ones found in timetabling problems.

The paper is structured as follows. First related papers are described in Section 8.2. The basic IP model is introduced in Section 8.3, including the essential constraints. Section 8.4 shows the TSD of this model, and derives the lower bound on room allocation penalties for the Stage I model. Section 8.5 extends the model so it encapsulates a practical version of the high school timetabling problem, defined by the online high school administration system Lectio. Section 8.6 shows computational results, comparing the decomposition to previous approaches for 100 problem instances taken from the Lectio database. Section 8.7 concludes on our findings.

8.2 Related work

Integer Programming has been used to model various educational timetabling problems. However, heuristics are still the most popular method for these problems, see surveys Schaerf (1999) and Pillay (2013). In terms of IP, de Werra (1985) describes what is called 'a simple model' for the class-teacher problem, and existence of solutions is proven under certain circumstances using graph theoretical models. The problems considered are feasibility problems, and soft constraints are not added to the models. Birbas et al. (1997) describes a 'fully defined' IP model for Greek secondary schools, which is evaluated on five different schools with success. Avella et al. (2007) formulates an IP model which is used to solve small instances of various origin. The IP is solved within a VLSN algorithm, with good results.

For the related university course timetabling problem, Daskalaki et al. (2004) presents a model which schedules courses to timeslots and classrooms, using many so called *operational rules*. Three different problem instances of significant size are all solved to optimality using CPLEX. MirHassani (2006) describes the problem for an Iranian university, and reports good results by applying the XA solver. In Dimopoulou and Miliotis (2001) an IP model is used to solve the timetabling problem for The Athens University of Economics and Business.

Decomposition of IP models for educational timetabling is not a very well researched topic. Burke et al. (2010) state that: *In the timetabling community, the "times first, rooms second" decomposition is a standard procedure.* However, it seems that this procedure has not been applied

much in context of IP models. Burke and Newall (1999) apply the procedure in context of an Evolutionary Algorithm for Examination Timetabling. In terms of multistage-decompositions, the importance of Lach and Lübbecke (2008) and Lach and Lübbecke (2012) has already been discussed. Carter (1983) presents an interesting decomposition algorithm for course timetabling with elective courses. Stating the problem in terms of a vertex coloring problem facilitates the decomposition of the graph by cliques, such that the subproblem defined by each clique is solved separately.

In Burke et al. (2010), experiments are conducted on disabling different combinations of soft-constraint penalties of the *Udine Course Timetabling Problem*, including one where all room penalties on room allocation are disabled. Thereby a similar decomposition to that of Lach and Lübbecke (2012) is obtained.

Daskalaki and Birbas (2005) presents an approach for university timetabling, where courses are first assigned to days (skipping some requirements for compactness), and in the following stage the timetable for each day is treated locally (enforcing the compactness). Convincing computational results are shown. In Birbas et al. (2009), a high school timetabling problem is solved by first allocating 'work shifts' to teachers, and then solving the actual timetabling problem. This is related to the type of decomposition performed in this paper. Badri (1996) uses a related approach for university course timetabling, where *faculties* are first assigned to courses, and then faculties are assigned to timeslots. However the problems solved are tiny.

Recently, high school timetabling received attention in the *International Timetabling Competition 2011* (ITC2011), see Post et al. (2012a). This competition built upon an uniform format for formulating problem instances (and their solutions), known as *XHSTT* (Post et al. (2012b)). Currently, around 50 problem instances are available in this format. The problem considered in this paper deviates from the XHSTT format in several important ways, which is beyond the scope of this section to elaborate on. Even though many researches participated in ITC2011, it seems that all were applying heuristics.

8.3 An Integer Programming Model for High School Timetabling

As the origin for our approach lies the IP model presented in Sørensen and Stidsen (2013). To make a clear presentation of the TSD, this IP model is reduced to its essential parts, which is described in the following. In Section 8.5, the full IP model is shown in context of the TSD.

A set of *events* $\mathcal{E}$ is given. Each event generally represents one lecture, which is defined as a meeting between specific *resources*, with a certain subject as teaching-objective. The set of resources is denoted $\mathcal{A}$. The goal of the high school timetabling problem is to assign each event to a room and to a timeslot, such that no conflicts among resources occur. The set of rooms and timeslots are denoted $\mathcal{R}$ and $\mathcal{T}$, respectively. The decision variable $x_{e,r,t} \in \{0,1\}$ takes value 1 if event $e \in \mathcal{E}$ is assigned room $r \in \mathcal{R}$ and timeslot $t \in \mathcal{T}$. To ensure a feasible solution exists, both the set of timeslots $\mathcal{T}$ and the set of rooms $\mathcal{R}$ are extended with a single dummy element, i.e. $\mathcal{T} = \{\mathcal{T} \cup t_D\}$ and $\mathcal{R} = \{\mathcal{R} \cup r_D\}$. This should be interpreted in the way that assigning to these dummy-elements actually means that no timeslot/room was assigned to the event. Thereby the goal of the IP is to assign as many events *as possible* to a timeslot and/or a room. From a practical point of view this is desirable, as the model is used in a decision support context where it might not be evident how to handle infeasibility. $\phi_{e,t} \in \mathbb{R}^+$ denotes the penalty for assigning event $e \in \mathcal{E}$ to timeslot $t \in \mathcal{T}$, and $\pi_{e,r} \in \mathbb{R}^+$ denotes the penalty for assigning event $e \in \mathcal{E}$ to

room $r \in \mathcal{R}$. Major penalties are given for assignments to the dummy-elements, i.e.

$$\phi_{e,t_D} \gg \phi_{e,t} \quad \forall e \in \mathcal{E}, t \in \mathcal{T} \setminus t_D \tag{8.1}$$
$$\pi_{e,r_D} \gg \pi_{e,r} \quad \forall e \in \mathcal{E}, r \in \mathcal{R} \setminus r_D \tag{8.2}$$

A room might be unavailable in certain timeslots, indicated by the binary parameter $G_{r,t} \in \{0,1\}$, which takes value 1 if room $r \in \mathcal{R}$ is available in timeslot $t \in \mathcal{T}$, and 0 otherwise. Furthermore, a set of eligible rooms exists for each event. Let parameter $K_{e,r} \in \{0,1\}$ take value 1 if event $e \in \mathcal{E}$ can take place in room $r \in \mathcal{R}$, and 0 otherwise. Each event requires a fixed set of resources. Let $E'_a, a \in \mathcal{A}$, denote the set of events where resource a participates.

We include in the model a set of constraints which will be described later, denoted by the constraint-set P_{other}, slightly abusing notation. These constraints define various other important criteria, such as forbidden timeslots for certain events, events which must be placed in the same timeslots, etc. Since these constraints are not required for describing the decomposition, their definitions are postponed to Section 8.5. We allow the set of constraints P_{other} to also denote soft-constraints (i.e. constraints which result in a weighted penalty in the objective function if it is not fulfilled). Thereby these constraints contain all necessary for conditions for modeling the timetabling instances in question, and represents a large set of distinct types of constraints. It will be argued in the next section that these constraints can be handled in the decomposition such that optimality of the IP model is not lost, with one exception.

Model (8.3) shows the IP model.

IP Model for High School Timetabling (8.3)

$$\min \quad w = \sum_{e \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{T}} (\phi_{e,t} + \pi_{e,r})\, x_{e,r,t} \tag{8.3a}$$

s.t.

$$\text{(one time/room)} \quad \sum_{r \in \mathcal{R}, t \in \mathcal{T}} x_{e,r,t} = 1 \qquad \forall e \in \mathcal{E} \tag{8.3b}$$

$$\text{(resource conf.)} \quad \sum_{r \in \mathcal{R}, e \in E'_a} x_{e,r,t} \leq 1 \qquad \forall a \in \mathcal{A}, t \in \mathcal{T} \setminus t_D \tag{8.3c}$$

$$\text{(room conf.)} \quad \sum_{e \in \mathcal{E}} x_{e,r,t} \leq G_{r,t} \quad \forall r \in \mathcal{R} \setminus r_D, t \in \mathcal{T} \setminus t_D \tag{8.3d}$$

$$\text{(eligible rooms)} \quad \sum_{t \in \mathcal{T}} x_{e,r,t} \leq K_{e,r} \quad \forall e \in \mathcal{E}, r \in \mathcal{R} \tag{8.3e}$$

$$x_{e,r,t} \in P_{\text{other}} \tag{8.3f}$$

$$x_{e,r,t} \in \{0,1\} \tag{8.3g}$$

The objective of the model is to minimize the overall penalty for assignments, given by (8.3a). Constraint (8.3b) specifies that each event must be assigned exactly one timeslot and one room. Events which require the same resource cannot be scheduled simultaneously (except in the dummy-timeslot), which is ensured by constraint (8.3c). A room cannot be used by more than one event in each timeslot. This is specified in constraint (8.3d). The requirement for eligible rooms is specified in constraint (8.3e). Constraint (8.3f) specifies that constraints P_{other} should be respected.

Theorem 2. *The High School Timetabling Problem as specified in* (8.3) *is $\mathcal{NP}$-hard.*

Proof. We conduct a reduction from Vertex Coloring. Let $G = (V, E)$ be an arbitrary graph and k be an arbitrary number. The question of the coloring problem would now be whether it is possible to color G with k colors, such that no two adjacent vertices share the same color.

Now construct a High School Timetabling instance in the following way:

- Let there be an event for every vertex, i.e. $\mathcal{E} = V$.
- Make sure there are enough rooms for all events, therefore create a room for every event (i.e. $|\mathcal{R}| = |\mathcal{E}|$) and make sure all events fit in all rooms (i.e. $K_{e,r} = 1 \, \forall e \in \mathcal{E}, r \in \mathcal{R}$), and that all rooms are available in all timeslots (i.e. $G_{r,t} = 1 \, \forall e \in \mathcal{E}, r \in \mathcal{R}$).
- For every edge $\{v_1, v_2\} \in E$ we create a resource in $\mathcal{A}$ (i.e. $\mathcal{A} = E$). The events using this resource will be the vertices connected by the edge (i.e. $E'_{\{v_1,v_2\}} = \{v_1, v_2\}$).
- We use exactly k timeslots (i.e. $\mathcal{T} = \{1, \ldots, k\}$).
- The additional constraints P_{other} can be dropped without loss of generality, as we impose no other restrictions on the timetabling instance. Also as we only search for a feasible solution, the soft constraints can easily be ignored.

Now we have a direct relation between the Vertex Coloring problem and the new timetabling instance. A solution of one problem can be transformed into a solution of the other by translating the colors of the vertices into timeslots, and vice versa. The rooms pose no restriction as every event can be scheduled in its own room.

Therefore solving the timetabling instance would result in solving the Vertex Coloring Problem, and the High School Timetabling problem is $\mathcal{NP}$-hard. □

Here we remark that Model (8.3) encapsulates many of the basic constraints required by most timetabling problems. If we for instance consider the XHSTT format, the basic requirement is to assign events to timeslots and resources (corresponding to rooms in our cases), subject to no clashes between resources. Therefore it is believed that the type of decomposition considered in this paper can in principle be applied to other timetabling problems as well.

8.4 Two-Stage Decomposition of the Integer Programming model

The TSD of Model (8.3) is performed as follows. The model is split into two stages; In Stage I, events are assigned to timeslots, and in Stage II, events are assigned to rooms. The respective decision variables for these stages are the following; $y_{e,t} \in \{0, 1\}$ takes value 1 if event $e \in \mathcal{E}$ is assigned timeslot $t \in \mathcal{T}$, and 0 otherwise; $z_{e,r} \in \{0, 1\}$ takes value 1 if event $e \in \mathcal{E}$ is assigned room $r \in \mathcal{R}$, and 0 otherwise. This means that constraints (8.3b) and (8.3c) are part of Stage I, and constraints (8.3d) and (8.3e) are part of Stage II.

As for constraints (8.3f), defined by the set P_{other}, it is assumed that each of the constraints in P_{other} is either only touching the assignment of events to timeslots (denoted P_{timeslot}) or the assignment of events to rooms (denoted P_{room}) It is shown in Section 8.5 that this assumption holds, with one exception. This means that constraints P_{timeslot} can be fully stated in terms of variable $y_{e,t}$, and constraints P_{room} can be fully stated in terms of variable $z_{e,r}$. As constraints P_{timeslot} are part of Stage I, these are handled optimally. This is different for P_{room}, as those constraints are harder to consider during Stage I. We will therefore address them with greater care in the next sections and show how we can add weighted room allocations to the decomposed model as a good approximation.

The solution obtained from the Stage I model is given as a parameter to the Stage II model, denoted $y^*_{e,t}$. The advantage of this approach is the huge reduction in the number of variables in

both stages, which results in a significantly decreased solving time. The following substitution of variables are made:

$$\sum_{r \in \mathcal{R}} x_{e,r,t} = y_{e,t} \quad \text{(Stage I)} \tag{8.4}$$

$$x_{e,r,t} = y^*_{e,t} z_{e,r} \quad \text{(Stage II)} \tag{8.5}$$

The objective (8.3a) of the original model defines a natural objective for both Stage I and Stage II, since it can be split into two independent expressions (denoted w^I and w^{II}, respectively). If this was not the case (e.g. if an event had different priorities for rooms depending on the timeslot it was assigned), it would complicate matters in terms of the Stage I model.

To sum up, Models (8.6) and (8.7) show Stage I and Stage II, respectively.

Stage I (8.6)

$$\min \quad w^I = \sum_{e \in \mathcal{E}, t \in \mathcal{T}} \phi_{e,t} y_{e,t} \tag{8.6a}$$

s.t.

$$\text{(one timeslot)} \quad \sum_{t \in \mathcal{T}} y_{e,t} = 1 \ \forall e \in \mathcal{E} \tag{8.6b}$$

$$\text{(resource conf.)} \quad \sum_{e \in E'_a} y_{e,t} \leq 1 \ \forall a \in \mathcal{A}, t \in \mathcal{T} \setminus t_D \tag{8.6c}$$

$$y_{e,t} \in P_{\text{timeslot}} \tag{8.6d}$$

$$y_{e,t} \in \{0,1\} \tag{8.6e}$$

Stage II (solution from Stage I is denoted $y^*_{e,t}$) (8.7)

$$\min \quad w^{II} = \sum_{e \in \mathcal{E}, r \in \mathcal{R}} \pi_{e,r} z_{e,r} \tag{8.7a}$$

s.t.

$$\text{(one room)} \quad \sum_{r \in \mathcal{R}} z_{e,r} = 1 \quad \forall e \in \mathcal{E} \tag{8.7b}$$

$$\text{(room conf.)} \quad \sum_{e \in \mathcal{E}} y^*_{e,t} z_{e,r} \leq G_{r,t} \ \forall r \in \mathcal{R} \setminus r_D, t \in \mathcal{T} \setminus t_D \tag{8.7c}$$

$$\text{(eligible rooms)} \quad z_{e,r} \leq K_{e,r} \ \forall e \in \mathcal{E}, r \in \mathcal{R} \tag{8.7d}$$

$$z_{e,r} \in P_{\text{room}} \tag{8.7e}$$

$$z_{e,r} \in \{0,1\} \tag{8.7f}$$

The outline of the TSD is shown in Figure 8.1. For a variable x the star-suffixed version x^* denotes a feasible solution. Stage I is solved using a MIP solver to obtain a solution $y^*_{e,t}$, which is given as input to the Stage II model. Note that Stage I possesses the coloring structure from theorem 2. Therefore Stage I is already a hard problem in its most basic form. Furthermore the value of the LP-relaxation of the Stage I model (denoted w^{I*}_{LP}) is a lower bound on the original model, as the Stage I model can be seen as a relaxation. Solving the Stage II model subject to the solution of the Stage I model obtains a solution $z^*_{e,t}$, and a solution to the original model $x^*_{e,r,t}$ can then be derived by equations (8.4) and (8.5).

For this paper we will solve Stage II using the specified IP even though we have not established its complexity yet. But as constructing a polynomial time algorithm for Stage II that can cope with all the additional constraints would be out of this papers scope we will postpone this to

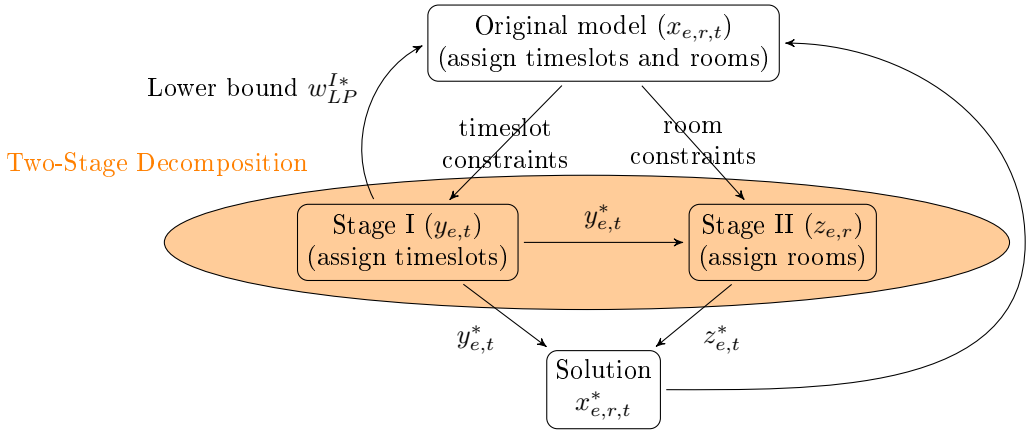


Figure 8.1: Two-Stage Decomposition flow chart.

a potentially later point in time. In the computational results section we will see that solving Stage II will not be the time-wise bottleneck anyhow.

As previously discussed, the penalties for room allocation can be implicitly handled in Stage I, which is described in Section 8.4.1. This extension of Stage I will not only allow better solutions to be found, but possibly also improvements in the bounds found by means of the LP relaxation.

As an alternative approach, we remark that an iterative procedure could in principle be used, such that the solution obtained from the Stage II model is given as input to the Stage I model, and the whole procedure is repeated. It is however unclear how the input from the Stage II model should effect the Stage I model to obtain convergence towards better solutions in terms of the overall objective. Furthermore, such an approach would require that both Stage I and Stage II can be solved 'quickly' (for the practical problem treated in this paper, computational results will show that this is in fact not the case for the Stage I model).

8.4.1 Extending Stage I with room allocation

The key idea behind extending Stage I with room allocation penalties is to consider Stage II as a matching problem in a bipartite graph. Constraints (8.7e) are set aside in the following, as they have not been defined yet. However, it will be seen later that these constraints does not fully obey the matching problem structure, and therefore Stage II must be solved with a MIP solver. This means that the room penalties are only partly incorporated in Stage I, but still this seems better than having Stage I being totally unaware of these penalties, as already discussed in Section 8.1.

Some basic graph notation is introduced in the following. A graph is *bipartite* if its set of vertices can be partitioned into two sets A and B, such that every edge has one endpoint in A, and the other endpoint in B. A *matching* in a graph is a set of edges such that no two of these edges share endpoints. A *maximum matching* is a matching that contains the largest possible number of edges. The *matching number* $\nu(G)$ of graph G is the number of edges in a maximum matching. For a graph with edge-weights, a *minimum weighted maximum matching* is a maximum matching where the sum of the weights on the edges of the matching is minimal.

In the Stage II model (8.7), notice first that the only constraint which treats timeslots is constraint (8.7c). Since this constraint applies to timeslots individually, Model (8.7) can be split into $|\mathcal{T}|$ independent optimization problems. Second, assume that the minimum weighted

maximum matching problem of the weighted bipartite graph $G_t = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E_t\right)$ fully describes the optimization problem of timeslot t of Model (8.7). To recognize this, let $\mathcal{R}_D$ be the set of $|\mathcal{E}|$ distinct dummy-rooms. I.e. for each event a dummy-room is created (and a corresponding edge is added to the graph) to ensure a matching of every event to a room will always exist. Hence the room-vertices of graph G_t is given by $\bar{\mathcal{R}} = \mathcal{R} \cup \mathcal{R}_D$. The set of edges is given by (skipping edge definitions for the dummy-room vertices) $E_t = \{e \in \mathcal{E}, r \in \mathcal{R} \mid K_{e,r} = 1 \wedge G_{r,t} = 1\}$, and the weight on each edge is given by $\pi_{e,r}$. The goal of the matching problem is to select a maximum matching with minimum weight. A trivial maximum matching will assign every event to the dummy-room. Clearly this resembles component $t \in \mathcal{T}$ of Model (8.7).

Stating the Stage II model in terms of this graph allows us to exploit some well-known properties of matching problems in bipartite graphs. In the following, notation is simplified by dropping the t-index where applicable, i.e. we write G instead of G_t and E instead of E_t. Denote by $\Gamma(S)$ the neighbors of event-nodes $S \subseteq \mathcal{E}$ in graph G, i.e. $\Gamma(S) = \left\{i \in \bar{\mathcal{R}} \mid j \in S, (i,j) \in E\right\}$. Hence $\Gamma(S) \subseteq \bar{\mathcal{R}}$. The well-known theorem of Hall states that a bipartite graph $G = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E\right)$ has a matching of all vertices $\mathcal{E}$ into $\bar{\mathcal{R}}$ if and only if $|\Gamma(S)| \geq |S| \ \forall S \subseteq \mathcal{E}$. Observe that for timeslot $t \in \mathcal{T}$, the variable $y_{e,t}$ determines whether event $e \in \mathcal{E}$ is part of graph G. Lach and Lübbecke (2012) used this theorem to add constraints of the form

$$\sum_{e \in S} y_{e,t} \leq |\Gamma(S)| \quad \forall S \subseteq \mathcal{E}, t \in \mathcal{T} \tag{8.8}$$

to the Stage I model to guarantee that the Stage I model would yield a feasible matching problem for every component $t \in \mathcal{T}$ of the Stage II model. However, such constraints are redundant in our case, as we are guaranteeing that no matter how $y_{e,t}$ is selected, a feasible matching will always exist (due to the dummy-rooms). Instead we modify the expression (8.8) to provide a lower bound on the weighted matching problem.

For the bipartite graph $G = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E\right)$ (the edge-weights are set aside for now), let the *deficiency* of a vertex set $S \subseteq \mathcal{E}$ be defined as $\operatorname{def}(S) = |S| - |\Gamma(S)|$. Let the deficiency of G be defined as $\operatorname{def}(G) = \max_{S \subseteq \mathcal{E}} \operatorname{def}(S)$. Theorem 1.3.1 of Lovász and Plummer (2009) states the following:

Theorem 3. *The matching number of the bipartite graph* $G = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E\right)$, *is* $\nu(G) = |\mathcal{E}| - \operatorname{def}(G)$.

I.e. for the bipartite graph G, $\operatorname{def}(G)$ denotes the amount of vertices which are not matched in the maximum matching.

Let $\mathcal{W}$ denote the ordered set of different values found in $\pi_{e,r}$, i.e.

$$\mathcal{W} = \left\{ w \in \mathbb{R}^+ \mid \exists e \in \mathcal{E}, \exists r \in \mathcal{R} : \pi_{e,r} = w \right\} \tag{8.9}$$

$$w_i < w_j \Leftrightarrow \operatorname{ord}(w_i) < \operatorname{ord}(w_j) \quad \forall (w_i, w_j) \in \mathcal{W} \tag{8.10}$$

Notice that no restrictions are posed on the amount of different values found, but it should be remarked that for our practical case, the cardinality of $\mathcal{W}$ is small (typically below 10).

The bipartite graph G is split into subgraphs, one subgraph for each $w \in \mathcal{W}$. A subgraph is denoted as $G_{\leq w} = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E_{\leq w}\right)$, where the set of edges are those with at least weight w,

$$E_{\leq w} = \{(e,r) \in E \mid \pi_{e,r} \leq w\} \tag{8.11}$$

By these definitions, it is clear that

$$|\Gamma(G_{\leq w_1})| \leq |\Gamma(G_{\leq w_2})| \leq \ldots \Rightarrow \tag{8.12}$$

$$\operatorname{def}(G_{\leq w_1}) \geq \operatorname{def}(G_{\leq w_2}) \geq \ldots \tag{8.13}$$

Using the deficiencies of these subgraphs, a lower bound on the minimum weight maximum matching can be stated. Let $a_w \in \mathbb{N}_0$ be defined as

$$a_{w_i} = \begin{cases} \nu\left(G_{\leq w_i}\right) - \nu\left(G_{\leq w_{i-1}}\right) & = \operatorname{def}\left(G_{\leq w_{i-1}}\right) - \operatorname{def}\left(G_{\leq w_i}\right) \quad \text{if } i > 1 \\ \nu\left(G_{\leq w_1}\right) & = |\mathcal{E}| - \operatorname{def}\left(G_{\leq w_1}\right) \quad \text{otherwise} \end{cases} \tag{8.14}$$

The intuition behind a_w is to measure the change in the matching number when edges with weight w are added to the subgraph $G_{\leq w_{i-1}}$. Note that $0 \leq a_w \leq |\mathcal{E}|$ for any $w \in \mathcal{W}$, as $0 \leq \operatorname{def}\left(G_{\leq w}\right) \leq |\mathcal{E}|$.

Theorem 4. *The quantity*

$$\sum_{w \in \mathcal{W}} w \cdot a_w$$

is a lower bound on a minimum weight maximum matching in the edge-weighted bipartite graph G.

Proof. Assume for contradiction there exists a maximum matching M with lower weight, i.e.

$$\sum_{e \in M} w_e < \sum_{w \in \mathcal{W}} w \cdot a_w$$

Let b_w denote the number of edges in M of weight lesser or equal w, i.e.

$$b_w = |\{e \in M : w_e \leq w\}|$$

Let k be the smallest number such that,

$$b_{w_k} > \sum_{i=1}^{k} a_{w_k}$$

This number must exist since M is a cheaper matching. For the subgraph $G_{\leq w_k}$, b_{w_k} can never exceed the matching number $\nu\left(G_{\leq w_k}\right)$ (by Theorem 3). We say 'exceed' as the matching might not include precisely $\nu\left(G_{\leq w_k}\right)$ edges of weight lesser or equal w_k. This gives:

$$\begin{aligned} b_{w_k} &\leq \nu\left(G_{\leq w_k}\right) \\ &= |\mathcal{E}| - \operatorname{def}\left(G_{\leq w_k}\right) \\ &= |\mathcal{E}| \underbrace{-\operatorname{def}\left(G_{\leq w_1}\right) + \operatorname{def}\left(G_{\leq w_1}\right) - \operatorname{def}\left(G_{\leq w_2}\right) + \ldots + \operatorname{def}\left(G_{\leq w_{k-1}}\right)}_{=0} - \operatorname{def}\left(G_{\leq w_k}\right) \\ &= \sum_{i=1}^{k} a_{w_i} \end{aligned}$$

which is a contradiction. □

This lower bound is minimized in the objective of the Stage I model. Hence, any lower bound on the Stage I model is a lower bound on the overall problem. Additional notation is needed for stating the extended Stage I model.

The neighbors of event-nodes $S \subseteq \mathcal{E}$ in graph $G_{t,\leq w}$ is denoted $\Gamma_{t,\leq w}(S)$ for timeslot t and weight w. Let the variable $\operatorname{def}_{t,\leq w} \in \mathbb{N}_0$ be the deficiency of subgraph $G_{t,\leq w}$. The deficiencies

for each subgraph can be determined by adding the following constraint (follows directly from Theorem 3 and the definition of the deficiency for a bipartite graph),

$$\sum_{e \in S} y_{e,t} - \text{def}_{t,\leq w} \leq |\Gamma_{t,\leq w}(S)| \quad \forall S \subseteq \mathcal{E}, t \in \mathcal{T}, w \in \mathcal{W} \tag{8.15}$$

Model (8.16) shows the extended model. Variables $\text{def}_{t,\leq w}$ and $a_{t,w}$ are specified to be continuous as they will naturally take integer values. Obviously an exponential amount of constraints is added due to (8.16d), but it will be shown that for our practical purpose, the amount of required constraints is low.

Stage I extended with Hall's condition (8.16)

$$\min \quad w^I = \sum_{e \in \mathcal{E}, t \in \mathcal{T}} \phi_{e,t} y_{e,t} + \sum_{t \in \mathcal{T}, w \in \mathcal{W}} w a_{t,w} \tag{8.16a}$$

s.t.

$$\text{(one timeslot)} \quad \sum_{t \in \mathcal{T}} y_{e,t} = 1 \qquad \forall e \in \mathcal{E} \tag{8.16b}$$

$$\text{(resource conf.)} \quad \sum_{e \in E'_a} y_{e,t} \leq 1 \qquad \forall a \in \mathcal{A}, t \in \mathcal{T} \setminus t_D \tag{8.16c}$$

$$\text{(Hall's)} \quad \sum_{e \in S} y_{e,t} - \text{def}_{t,\leq w} \leq |\Gamma_{t,\leq w}(S)| \quad \forall S \subseteq \mathcal{E}, t \in \mathcal{T}, w \in \mathcal{W} \tag{8.16d}$$

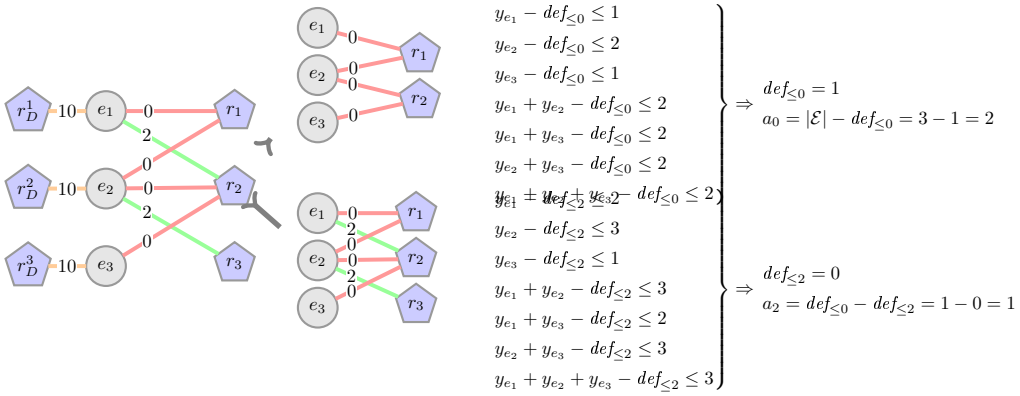
$$\text{(LB)} \quad |\mathcal{E}| - \text{def}_{t,\leq w_1} = a_{t,w_1} \qquad \forall t \in \mathcal{T} \tag{8.16e}$$

$$\text{(LB)} \quad \text{def}_{t,\leq w_{-1}} - \text{def}_{t,\leq w} = a_{t,w} \qquad \forall t \in \mathcal{T}, w \in \mathcal{W}, \text{ord}(w) > 1 \tag{8.16f}$$

$$y_{e,t} \in \{0,1\} \tag{8.16g}$$

$$\text{def}_{t,\leq w}, a_{t,w} \in \mathbb{R}^+ \tag{8.16h}$$

Example 1. *Below is shown an example of a bipartite graph and its subgraphs for some timeslot. Three different room-weights exists, $\mathcal{W} = \{0, 2, 10\}$. Clearly, an optimal solution to the matching problem of this graph is $(e_1, r_1), (e_2, r_3), (e_3, r_2)$ with value 2. The subgraphs for weights 0 and 2 are shown, and the lower bound derived.*



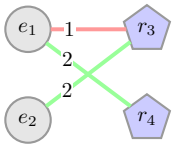
$$\left.\begin{aligned}
&y_{e_1} - def_{\leq 0} \leq 1\\
&y_{e_2} - def_{\leq 0} \leq 2\\
&y_{e_3} - def_{\leq 0} \leq 1\\
&y_{e_1} + y_{e_2} - def_{\leq 0} \leq 2\\
&y_{e_1} + y_{e_3} - def_{\leq 0} \leq 2\\
&y_{e_2} + y_{e_3} - def_{\leq 0} \leq 2\\
&y_{e_1} + y_{e_2} + y_{e_3} - def_{\leq 0} \leq 2
\end{aligned}\right\} \Rightarrow \begin{aligned} &def_{\leq 0} = 1\\ &a_0 = |\mathcal{E}| - def_{\leq 0} = 3 - 1 = 2 \end{aligned}$$

$$\left.\begin{aligned}
&y_{e_2} - def_{\leq 2} \leq 3\\
&y_{e_3} - def_{\leq 2} \leq 1\\
&y_{e_1} + y_{e_2} - def_{\leq 2} \leq 3\\
&y_{e_1} + y_{e_3} - def_{\leq 2} \leq 2\\
&y_{e_2} + y_{e_3} - def_{\leq 2} \leq 3\\
&y_{e_1} + y_{e_2} + y_{e_3} - def_{\leq 2} \leq 3
\end{aligned}\right\} \Rightarrow \begin{aligned} &def_{\leq 2} = 0\\ &a_2 = def_{\leq 0} - def_{\leq 2} = 1 - 0 = 1 \end{aligned}$$

Hence the lower bound is derived as:

$$LB = a_0 0 + a_2 2 = 2$$

Example 2. *Naturally, the lower bound is not necessarily tight, as shown by the following small example.*



For weight $w_1 = 1$ *the deficiency of* G_1 *is* $\mathrm{def}(G_1) = 1$ *and therefore* $a_1 = 1$. *As the deficiency for* G_2 *is* $\mathrm{def}(G_2) = 0$ *we also have* $a_2 = 1$. *The lower bound therefore reads* $1 \cdot 1 + 2 \cdot 1 = 3$. *But obviously the only (and therefore minimal weight) maximum matching has weight* 4. *In fact, by increasing the weight on the weight* 2 *edges, it is seen that the gap between the lower bound and the actual minimal weight maximum matching could potentially be arbitrarily large. For the practical problem handled later, the weights can take values* $\{1, 2, \ldots, 10\}$, *and therefore the gap between weights is low. The gap between the lower bound and the actual matchings obtained will be investigated experimentally.*

8.4.2 An exact approach using Egerváry's theorem

An an alternative approach to the derived lower bound, the theorem of Egerváry (Egerváry (1931)) can be used to characterize the minimum weight of a matching in a bipartite graph, which deserves a mentioning in this context. The theorem states the following ((Schrijver, 2003, Theorem 17.1), here stated as a minimum weight problem):

Theorem 5. *Let* $G = (V, E)$ *be a bipartite graph and let* $w : E \to \mathbb{R}^+$ *be a weight function. Then the minimum weight of a matching in* G *is equal to the maximum value of* $y(V)$, *where* $y : V \to \mathbb{R}^+$ *is such that*

$$y_u + y_v \leq w_e \quad \forall u, v \in V, (u, v) \in E$$

However, since we consider a bipartite graph for each timeslot, and each bipartite graph in worst case has $|\mathcal{E}|\,|\mathcal{R}|$ vertices (which occurs often in practice), the amount of required constraints is of magnitude $|\mathcal{E}|\,|\mathcal{R}|\,|\mathcal{T}|$, so this is not a tractable approach. Furthermore, since the graph is not static (i.e. its structure depends on assignments of events to timeslots), a $\min - \max$ formulation would be required.

8.4.3 Generating Hall's inequalities

To generate the Hall's inequalities, it is necessary to exploit the structure of the underlying graph. I.e. we use problem specific knowledge to overcome the requirement of enumerating all subsets of events. In the Lectio case, two important features are known about the bipartite graphs:

- An event often has a special association with one specific room. This is either because the event is locked to that room, or because a penalty is imposed on *not* assigning an event to the room it was previously assigned to. In the later case, this means that one room has a lower weight than all other rooms for the particular event. Hence, in the subgraph $G_{\leq w}$ for this respective lower weight, only a single edge exists for the event. An event with only a single adjacent edge is denoted as a *singleton* event from now on.
- Furthermore, predefined feature-groups of rooms exist. A feature group of rooms is devoted to a certain type of lecture, for instance chemistry or physics, which require a room with specialized equipment. Hence many events are adjacent to the exact same set of rooms.

These graphs are hence exploited by separately considering the inequalities induced by singleton events and the inequalities induced by all other events, and finally those inequalities induced by combining these. The approach is formalized below. It should be remarked that applying this type of decomposition will require exploiting at least some properties of the underlying graph. We refer to Balas and Pulleyblank (1983), Edmonds (1965) and Lach and Lübbecke (2008) as helpful resources in this aspect.

For a subset of rooms $R \subseteq \mathcal{R}$, let $\Gamma^{-1}(R)$ be the set of events adjacent to *only* rooms in R, i.e. $\Gamma^{-1}(R) = \{ e \in \mathcal{E} \mid \Gamma(\{e\}) \subseteq R \}$.

Theorem 6. *The Hall inequalities*

$$\sum_{e \in S} y_e - \text{def} \leq |\Gamma(S)| \qquad \forall S \subseteq \mathcal{E}$$

are fully contained in

$$\sum_{e \in \Gamma^{-1}(R)} y_e - \text{def} \leq |R| \qquad \forall R \subseteq \mathcal{R}$$

Proof. Let $S \subseteq \mathcal{E}$ be any set of events. Now we let $R = \Gamma(S)$. Obviously we have $S \subseteq \Gamma^{-1}(R) = \Gamma^{-1}(\Gamma(S))$. If $\sum_{e \in \Gamma^{-1}(R)} y_e - \text{def} \leq |R|$ holds we get

$$\sum_{e \in S} y_e - \text{def} \leq \sum_{e \in \Gamma^{-1}(R)} y_e - \text{def} \leq |R| = |\Gamma(S)| \qquad \square$$

This means that instead of having a constraint for every subset of events we can do with a constraint for every subset of rooms (which are considerably less).

Next we can further reduce the number of necessary constraints by exploiting symmetry between rooms. Rooms which are adjacent to exactly the same events can be grouped, and essentially treated as one room.

Theorem 7. *Let $R_1, \ldots, R_m \subseteq \mathcal{R}$ be distinct ($i \neq j \Rightarrow R_i \cap R_j = \emptyset$) subsets of rooms. Let $I \subseteq \{1, \ldots, m\}$ be an index set such that*

$$\bigcup_{i \in I} \Gamma^{-1}(R_i) = \Gamma^{-1}\left(\bigcup_{i \in I} R_i \right) \qquad \textit{(i.e. there is no event that only fits into a combination of the room sets in } I\textit{)}$$

Then the Hall constraint

$$\sum_{e \in \Gamma^{-1}\left(\bigcup_{i \in I} R_i\right)} y_e - \text{def} \leq \left| \bigcup_{i \in I} R_i \right|$$

is dominated by

$$\sum_{e \in R_i} y_e - \text{def}_i \leq |R_i| \qquad \forall i \in I$$
$$\sum_{i \in I} \text{def}_i \qquad \leq \text{def}$$

where $\text{def}_i \in \mathbb{N}_0$ *is the deficiency of index* $i \in I$*, i.e.* $\text{def}_i = \text{def}\left(\Gamma^{-1}(R_i)\right)$.

Proof. First note that $\sum_{i\in I} \mathrm{def}_i \leq \mathrm{def}$ implies

$$\sum_{e\in\Gamma^{-1}\left(\bigcup_{i\in I} R_i\right)} y_e - \mathrm{def} \leq \sum_{i\in I} \sum_{e\in\Gamma^{-1}(R_i)} y_e - \mathrm{def}_i$$

Next by $\sum_{e\in\Gamma^{-1}(R_i)} y_e - \mathrm{def}_i \leq |R_i|$ we get

$$\sum_{i\in I} \sum_{e\in\Gamma^{-1}(R_i)} y_e - \mathrm{def}_i \leq \sum_{i\in I} |R_i| = \left|\bigcup_{i\in I} R_i\right|$$

where the later equality holds as the R_i are distinct. □

Since the amount of possible ways to select I is exponential, this shows a potential way to limit the amount of necessary inequalities.

The graphs of the Lectio instances usually have the following structure, as previously discussed: Certain events are fixed to a specific room. These events are known as singleton events, and are denoted with the set E^1. If the singleton events are discarded, all other rooms can be grouped into groups $\mathcal{I} = \{1, 2, \ldots, m\}$, i.e. $R_i \subseteq \mathcal{R}\ \forall i \in \mathcal{I}$, where every room is connected to the very same events as the other rooms of the same group. In particular this means

$$\Gamma^{-1}(R) \setminus E^1 = \emptyset \quad \forall R \subsetneq R_i, i \in \mathcal{I} \tag{8.17}$$

I.e. no event is adjacent to only a subset of rooms of the room-groups, except for the singleton events. The key observation here is that the number of these groups of rooms is low, yielding a tractable way to generate the Hall inequalities. By Theorem 6 we know that a subset of rooms fully characterises one of the Hall constraints (and that it is sufficient to consider only those).

Corollary 1. *Given the structure of the Lectio graphs, only the following subsets of rooms need to be considered wrt. eq.* (8.15) *(in the altered form defined by Theorem 6):*

$$(I) \quad \Gamma(e) \qquad \forall e \in E^1 \tag{8.18}$$

$$(II) \quad \bigcup_{i\in I} R_i \quad \forall I \subseteq \mathcal{I} \tag{8.19}$$

Proof. For contradiction, let $\tilde{R} \subseteq \mathcal{R}$ be any other subset of rooms, i.e.

$$\tilde{R} \neq \Gamma(e) \quad \forall e \in E^1$$
$$\tilde{R} \neq \bigcup_{i\in I} R_i \quad \forall I \subseteq \mathcal{I}$$

$\tilde{R}$ can be decomposed into subsets $\tilde{R}_i, i \in \mathcal{I}$, such that $\tilde{R}_1 \subseteq R_1, \tilde{R}_2 \subseteq R_2, \ldots, \tilde{R}_m \subseteq R_m$ and $\bigcup_{i\in\mathcal{I}} \tilde{R}_i = \tilde{R}$.

For each of the decomposed room sets $\tilde{R}_i$ we can now have one of the three following cases (by eq. (8.17), which disallows that $\tilde{R}_i \neq R_i$ and $\Gamma^{-1}\left(\tilde{R}_i\right) \setminus E^1 \neq \emptyset$):

1. $\tilde{R}_i = R_i$
2. $\tilde{R}_i \neq R_i$ and $\Gamma^{-1}\left(\tilde{R}_i\right) \cap E^1 \neq \emptyset$
3. $\tilde{R}_i \neq R_i$ and $\Gamma^{-1}\left(\tilde{R}_i\right) = \emptyset$

Let the respective indices be contained in the sets I_1, I_2 and I_3. The rooms from the third case $(\tilde{R}_i, i \in I_3)$ do not add events to the left hand side of a Hall constraint and can therefore be ignored.

If combining the rooms from the first case to $R' = \bigcup_{i \in I_1} \tilde{R}_i$ we get one of the already considered combinations of room groups. Now note that there is no event fitting into the combination of R' with any of the rooms from the second case (their Γ^{-1} only contains singleton events) and therefore the condition for Theorem 7 is met:

$$\left(\bigcup_{i \in I_2} \Gamma^{-1}\left(\tilde{R}_i\right) \right) \cup \Gamma^{-1}\left(R'\right) = \Gamma^{-1}\left(\left(\bigcup_{i \in I_2} \tilde{R}_i \right) \cup R' \right)$$

So we now know that the Hall constraint corresponding to $\tilde{R}$ is unnecessary. □

Algorithm (1) shows the implemented algorithm for generation all necessary Hall constraints according to this construction.

Algorithm 1 Generating Hall's conditions

1: **input:** bipartite graph $G = \left(\mathcal{E} \cup \bar{\mathcal{R}}, E\right)$
2: **output:** set of sets of rooms H, which each constitute a Hall inequality
3: identify E^1 of G
4: $N_r = \left\{ e \in \mathcal{E} \setminus E^1 \mid r \in \Gamma\left(e\right) \neq \emptyset \right\}$ ▷ Identity adjacent events for each room
5: $T = \left\{ R \subseteq \bar{\mathcal{R}} \mid (r_i, r_j) \in R, i \neq j, N_{r_i} = N_{r_j} \right\}$ ▷ Groups of rooms which are adjacent to the same events
6: **for all** $S \in \mathcal{P}\left(T\right)$ **do** ▷ $\mathcal{P}\left(T\right)$ denotes the powerset of T, i.e. the collection of all subsets
7: $H = H \cup \left\{ r \in R \mid R \in S \right\}$ ▷ Add set of room (eq. (8.19))
8: **end for**
9: $H = H \cup \{r\} \quad \forall r \in \Gamma\left(E^1\right)$ ▷ Add rooms of singleton events (eq. (8.18))

In Line 5 rooms are grouped. Here it should be remarked that this is done in a way that identifies the minimum number of groups of rooms. The amount of generated inequalities is hence exponential in the number of groups of rooms. For the Lectio high school timetabling problem, this number is in general low. However, an artificial limit of a maximum of 12 different groups of rooms is imposed, allowing in magnitude of 2^{12} inequalities to be generated for each timeslot. In practice, only two problem instances are restricted by this limit ("HasserG2012" and "SlagelG2012"). The room groups to generate inequalities are selected by ordering the room groups in terms of total number of adjacent events to all other room groups, and taking those room groups where this number is highest. Obviously, omitting some inequalities will not change the fact that the room allocation penalty added to Stage I is a lower bound on the objective of Stage II.

8.5 Lectio High School Timetabling Problem

To establish computational results, Stage I and Stage II are extended to the full version of the Danish case of high school timetabling described in Sørensen and Stidsen (2013). This variant of the problem is used in the timetabling component of the high school ERP-system Lectio, and hence reflects all aspects of a practical timetabling optimization problem. Lectio Timetabling is used by many high schools in Denmark, and this formulation of the problem has been used in production mode for over a year. In this paper a brief introduction to each of the added

constraints and variables is given. More in-depth description and motivation can be found in Sørensen and Stidsen (2013).

This timetabling problem contains more types of constraints than what is usually found in the literature. This is mainly related to the big number of different high schools which use it, which inevitably gives a big variety of required features. However, a conversion scheme from this timetabling problem to the general XHSTT format is known, so the Lectio problem fits within the general concepts of high school timetabling problems.

Extensive computational experiments have shown that the usual formulation of this timetabling problem using a binary variable with three indices is very challenging for the commercial MIP solver Gurobi 5, which is among the very best general-purpose MIP solvers according to recent benchmarks of Mittelman (2013). Therefore this timetabling problem is a good candidate for testing the TSD approach.

8.5.1 Stage I

The set of timeslots $\mathcal{T}$ is defined by the combination of the set of days $\mathcal{D}$, and the set of daily-timeslots (known as *modules*) $\mathcal{M}$. The set of resources $\mathcal{A}$ consists of teachers and students, which is also known as the set of *entities*. Furthermore, the set of *classes* is denoted $\mathcal{C}$. A class $c \in \mathcal{C}$ is a *non-physical* resource treating a specific teaching-subject, and is associated with a certain set of events. Hence, an event can be viewed as a single lecture of a certain class. Parameter $J_{e,c} \in \{0,1\}$ takes value 1 if event $e \in \mathcal{E}$ is part of class $c \in \mathcal{C}$, and 0 otherwise.

Variable $v_{a,t} \in \{0,1\}$ takes value 1 if entity $a \in \mathcal{A}$ is active in timeslot $t \in \mathcal{T}$, and 0 otherwise. Variable $f_{a,d} \in \{0,1\}$ takes value 1 if entity $a \in \mathcal{A}$ has no events scheduled on events on day $d \in \mathcal{D}$ (we say that the entity has a *day off*, even though he/she might be occupied by unscheduled activities, such as lecture preparation), and 0 otherwise. Variable $b_{c,t} \in \{0,1\}$ takes value 1 if class $c \in \mathcal{C}$ has at least one lecture in timeslot $t \in \mathcal{T}$, and 0 otherwise. Variable $n_{c,d} \in \{0,1\}$ takes value if class $c \in \mathcal{C}$ has a neighbor-day conflict on day $d \in \mathcal{D}$. A neighbor-day conflict occurs when the same class has scheduled events on two consecutive days. Variable $o_{a,d} \in \{0,1\}$ takes value 1 if entity $a \in \mathcal{A}$ has only one event on day $d \in \mathcal{D}$, and 0 otherwise. Days with only one lecture are undesirable and should be avoided. Variable $w_c \in \mathbb{N}_0$ is the amount which class $c \in \mathcal{C}$ is 'out of week-balance'. I.e. if the set of timeslots is made up of times from more than one week, the amount of events of each class in each week must be equivalent (as far as possible). Variable $h_{a,d} \in \mathbb{N}_0$ is the amount of idle timeslots (a timeslot with no activity, but there exists both at least one earlier and one later timeslot with activity) for entity $a \in \mathcal{A}$ on day $d \in \mathcal{D}$. Variables $\underline{h}_{a,d}, \overline{h}_{a,d} \in \mathbb{N}_0$ denote the ordinal number of the first and last timeslot with activity on day $d \in \mathcal{D}$ for entity $a \in \mathcal{A}$, respectively.

The objective consists of 6 additional terms. These denote the weighted sum of entity idle slots (weight $\phi_a \in \mathbb{R}^+$), neighbor-day conflicts (weight $\zeta \in \mathbb{R}^+$), days with only one lecture (weight $\eta_a \in \mathbb{R}^+$), days-off for teachers (weight $\gamma_a \in \mathbb{R}^+$), days-off for students (weight $\delta_a \in \mathbb{R}^+$), and class week stability (weight $\iota \in \mathbb{R}^+$), respectively.

Let parameters S_e and C_e be the set of events which should be scheduled in the same timeslot as event $e \in \mathcal{E}$, and in the timeslot immediately following event $e \in \mathcal{E}$, respectively. Parameter $P_{d,d'} \in \{0,1\}$ takes value 1 if day $d \in \mathcal{D}$ and day $d' \in \mathcal{D}$ are neighbor-days, and 0 otherwise. Parameter $R_{c,d} \in \{0,1\}$ takes value 1 if class $c \in \mathcal{C}$ is part of some event which is locked to some timeslot on day $d \in \mathcal{D}$, and let $N_c \in \mathbb{N}_0$ be the number of allowed neighbor-day conflicts for class $c \in \mathcal{C}$. $\mathcal{T}_d$ denotes the set of timeslots on day $d \in \mathcal{D}$. Parameter $D_{e,t} \in \{0,1\}$ takes value 1 if event $e \in \mathcal{E}$ can be scheduled in timeslot $t \in \mathcal{T}$, and 0 otherwise. Parameter $F_a \in \mathbb{N}_0$ denotes the amount of required days off for entity $a \in \mathcal{A}$. Parameter $W_a \in \mathbb{N}_0$ denotes the maximum amount of events which can be scheduled to entity $a \in \mathcal{A}$ on any given day.

A class can only have one event assigned to each day, unless it is specified that multiple events should be placed in contiguous positions. We say that such *day-conflicts* are infeasible. The set $E''' \subseteq \mathcal{E}$ denotes the set of events for which day-conflicts are checked.

The most common case is that a school creates a timetable for a single week. However, some schools desire to create a two-week timetable instead. This allows more flexibility in the planning; take for instance a class with a nominated teaching load of three events per week. In case the school uses a two-week timetable, this class can for instance have one double lecture in the first week, and two double lectures in the second week. $d\left(\underline{\mathcal{T}}\right)$ and $d\left(\overline{\mathcal{T}}\right)$ denotes the set of days in the first and in the second week, respectively. $\underline{\mathcal{T}}$ and $\overline{\mathcal{T}}$ denotes the timeslots in the first and second week, respectively.

The complete Stage I model is shown in (8.20).

Stage I Lectio (8.20)

$$\begin{aligned}\min \quad w^I = \; & \sum_{e\in\mathcal{E},t\in\mathcal{T}} \phi_{e,t} y_{e,t} + \sum_{t\in\mathcal{T},w\in\mathcal{W}} w a_{t,w} + \sum_{a\in\mathcal{A},d\in\mathcal{D}} \beta_a h_{a,d} \\ & + \zeta \sum_{c\in\mathcal{C},d\in\mathcal{D}} n_{c,d} + \sum_{a\in\mathcal{A},d\in\mathcal{D}} \eta_a o_{a,d} + \sum_{a\in\mathcal{A}} \gamma_a \left[|\mathcal{D}| - \sum_{d\in\mathcal{D}} f_{a,d} \right] \\ & + \sum_{a\in\mathcal{A},d\in\mathcal{D}} \delta_a f_{a,d} + \iota \sum_{c\in\mathcal{C}} w_c \end{aligned} \tag{8.20a}$$

s.t.

(one timeslot)	$\sum_{t\in\mathcal{T}} y_{e,t}$	$= 1$	$\forall e \in \mathcal{E}$	(8.20b)
(entity time aux.)	$\sum_{e\in E'_a} y_{e,t}$	$= v_{a,t}$	$\forall a \in \mathcal{A}, t \in \mathcal{T}$	(8.20c)
(entity conf.)	$\sum_{t\in\mathcal{T}_d} v_{a,t} + f_{a,d}$	≤ 1	$\forall a \in \mathcal{A}, d \in \mathcal{D}$	(8.20d)
(Hall's)	$\sum_{e\in S} y_{e,t} - \mathrm{def}_{t,\leq w} - \lvert\Gamma_{t,\leq w}(S)\rvert$	≤ 0	$\forall S \subseteq \mathcal{E}, t \in \mathcal{T}, w \in \mathcal{W}$	(8.20e)
(room alloc. lb)	$\lvert\mathcal{E}\rvert - \mathrm{def}_{t,\leq w_1}$	$= a_{t,w_1}$	$\forall t \in \mathcal{T}$	(8.20f)
(room alloc. lb)	$\mathrm{def}_{t,\leq w_{-1}} - \mathrm{def}_{t,\leq w}$	$= a_{t,w}$	$\forall t \in \mathcal{T}, w \in \mathcal{W}, \mathrm{ord}(w) > 1$	(8.20g)
(locked time)	$y_{e,t}$	$= 1$	$\forall e \in \mathcal{E}, t \in \mathcal{T}, LT_{e,t} = 1$	(8.20h)
(same time)	$y_{e,t} - y_{e',t}$	$= 0$	$\forall e \in \mathcal{E}, e' \in S_e, t \in \mathcal{T}$	(8.20i)
(cont. times)	$y_{e,t} - y_{e',t'}$	$= 0$	$\forall e \in \mathcal{E}, e' \in C_e, (t,t') \in \mathcal{T}, d_t = d_{t'}, \quad \mathrm{ord}(t)+1 = \mathrm{ord}(t')$	(8.20j)
(n.d. conf.)	$\sum_{t\in\mathcal{T}_d} b_{c,t} + \sum_{t\in\mathcal{T}_{d'}} b_{c,t} - n_{c,d}$	≤ 1	$\forall c \in \mathcal{C}, (d,d') \in \mathcal{D}, P_{d,d'} = 1, R_{c,d} + R_{c,d'} \leq 1$	(8.20k)
(n.d. conf.)	$\sum_{d\in\mathcal{D}} n_{c,d}$	$\leq N_c$	$\forall c \in \mathcal{C}$	(8.20l)
(forbid. times)	$\sum_{t\in\mathcal{T},D_{e,t}=0} y_{e,t}$	$= 0$	$\forall e \in \mathcal{E}$	(8.20m)
(idle slots)	$\overline{h}_{a,d} - \underline{h}_{a,d} - \sum_{t\in\mathcal{T}_d} v_{a,t} + 1$	$= h_{a,d}$	$\forall a \in \mathcal{A}, d \in \mathcal{D}$	(8.20n)
(idle slots)	$\lvert\mathcal{M}\rvert - (\lvert\mathcal{M}\rvert - \mathrm{ord}(t))\, v_{a,t}$	$\geq \underline{h}_{a,d}$	$\forall a \in \mathcal{A}, d \in \mathcal{D}, t \in \mathcal{T}_d$	(8.20o)
(idle slots)	$\mathrm{ord}(t) v_{a,t}$	$\leq \overline{h}_{a,d}$	$\forall a \in \mathcal{A}, d \in \mathcal{D}, t \in \mathcal{T}_d$	(8.20p)
(days off)	$\sum_{d\in\mathcal{D}} f_{a,d}$	$\geq F_a$	$\forall a \in \mathcal{A}$	(8.20q)
(days off)	$\sum_{t\in\mathcal{T}_d} v_{a,t} + f_{a,d}$	≥ 1	$\forall a \in \mathcal{A}, d \in \mathcal{D}$	(8.20r)
(day conf.)	$\sum_{e\in E'''} J_{e,c} y_{e,t}$	$\leq b_{c,t}$	$\forall c \in \mathcal{C}, t \in \mathcal{T}$	(8.20s)
(day conf.)	$\sum_{t\in\mathcal{T}_d} b_{c,t}$	≤ 1	$\forall c \in \mathcal{C}, d \in \mathcal{D}$	(8.20t)
(work limit)	$\sum_{e\in\mathcal{E},t\in\mathcal{T}_d} y_{e,t}$	$\leq W_a$	$\forall a \in \mathcal{A}, d \in \mathcal{D}$	(8.20u)

$$\text{(one lecture)} \quad 2 - \sum_{t \in \mathcal{T}_d} v_{a,t} - 2f_{a,d} \leq o_{a,d} \quad \forall a \in \mathcal{A}, d \in \mathcal{D} \tag{8.20v}$$

$$\text{(class stabl.)} \quad \Big| \sum_{e \in \mathcal{E}, t \in \underline{\mathcal{T}}} J_{e,c} y_{e,t} - \sum_{e \in \mathcal{E}, t \in \overline{\mathcal{T}}} J_{e,c} y_{e,t} \Big| - 1 = w_c \quad \forall c \in \mathcal{C} \tag{8.20w}$$

$$\text{(d.o. stabl.)} \quad \Big| \sum_{d \in d(\underline{\mathcal{T}})} f_{a,d} - \sum_{d \in d(\overline{\mathcal{T}})} f_{a,d} \Big| \leq 1 \quad \forall a \in \mathcal{A} \tag{8.20x}$$

$$y_{e,t} \in \{0,1\} \tag{8.20y}$$

$$v_{a,t}, f_{a,d}, b_{c,t}, n_{c,d}, o_{a,d} \in [0,1] \tag{8.20z}$$

$$h_{a,d}, \underline{h}_{a,d}, \overline{h}_{a,d}, w_c, \text{def}_{t,\leq w}, a_{t,w} \in \mathbb{R}^+ \tag{8.20aa}$$

Constraint (8.20c) constraints the auxiliary variable $v_{a,t}$ properly. Constraint (8.20d) treats entity conflicts in a slightly changed formulation, to also constrain variable $f_{a,d}$ properly. Constraints (8.20e)-(8.20g) define the lower bound on room allocation, and are similar to those previously described. Constraint (8.20h) ensures the assigning of events are locked to a certain timeslot. Constraints (8.20i) and (8.20j) ensure the placement of events which must be placed in the same/contiguous timeslots. Constraints (8.20k) and (8.20l) ensure that variable $n_{c,d}$ is constrained properly, and that no more than N_c neighbor-day conflicts are scheduled for class $c \in \mathcal{C}$. Constraint (8.20m) poses restrictions on timeslots for which an entity is unavailable. Constraints (8.20n)-(8.20p) ensures that idle slots for entities are penalized accordingly. Constraint (8.20q) ensures that sufficient days off is assigned to each entity. Constraint (8.20r) makes sure that if an entity $a \in \mathcal{A}$ has no event on some day $d \in \mathcal{D}$, then variable $f_{a,d}$ is forced to take value 1. This is necessary as this variable is minimized in the objective. Constraints (8.20s) and (8.20t) ensure that day-conflicts for classes does not occur, and constraints the variable $b_{c,t}$ properly. Constraint (8.20u) ensures that the limit on the amount of events assigned to a day for entity a is respected. For an entity, days with only one event scheduled are undesirable. Constraint (8.20v) penalizes days with only one events scheduled for entity a. Constraint (8.20w) forces week-stability for events of classes, i.e. in case two-weeks are being planned, events for courses must be spread evenly throughout the weeks. Constraint (8.20x) ensures that in case several weeks are being planned, the days off for an entity are spread evenly throughout the weeks.

8.5.2 Stage II

Let variable $v_{c,r} \in \{0,1\}$ take value 1 if there is at least one event of which class $c \in \mathcal{C}$ participates assigned to room $r \in \mathcal{R}$, and 0 otherwise. Variable $s_c \in \mathbb{N}_0$ is the amount of 'excess' rooms assigned to events of class $c \in \mathcal{C}$, i.e. the total amount of rooms assigned minus one. This is used to enforce room stability for classes, since it is undesirable for a class to be assigned too many different rooms. Parameter $LR_{e,r} \in \{0,1\}$ takes value 1 if event $e \in \mathcal{E}$ is locked room $r \in \mathcal{R}$.

Stage II Lectio (8.21)

$$\min \quad w^{II} = \sum_{e \in \mathcal{E}, r \in \mathcal{R}} \pi_{e,r} z_{e,r} + \epsilon \sum_c s_c \tag{8.21a}$$

s.t.

$$\text{(one room)} \quad \sum_{r \in \mathcal{R}} z_{e,r} = 1 \quad \forall e \in \mathcal{E} \tag{8.21b}$$

$$\text{(room conf.)} \quad \sum_{e \in \mathcal{E}} y^*_{e,t} z_{e,r} \leq G_{r,t} \quad \forall r \in \mathcal{R} \setminus r_D, t \in \mathcal{T} \setminus t_D \tag{8.21c}$$

$$\text{(eligible rooms)} \quad z_{e,r} \leq K_{e,r} \quad \forall e \in \mathcal{E}, r \in \mathcal{R} \tag{8.21d}$$

$$\text{(locked rooms)} \quad z_{e,r} = 1 \quad \forall e \in \mathcal{E}, r \in \mathcal{R}, LR_{e,r} = 1 \tag{8.21e}$$

$$\text{(room stbl.)} \quad \sum_{e \in \mathcal{E}, t \in \mathcal{T} \setminus t_D} J_{e,c} y^*_{e,t} z_{e,r} - \sum_{e \in \mathcal{E}} J_{e,c} v_{c,r} \leq 0 \quad \forall r \in \mathcal{R} \setminus r_D, c \in \mathcal{C} \tag{8.21f}$$

$$\text{(room stbl.)} \quad \sum_{r \in \mathcal{R}} v_{c,r} - 1 \leq s_c \quad \forall c \in \mathcal{C} \tag{8.21g}$$

$$\text{(not only room)} \quad \sum_{r \in \mathcal{R} \setminus r_D} y^*_{e,t_D} z_{e,r} - \sum_{r \in \mathcal{R}} LR_{e,r} \leq 0 \quad \forall e \in \mathcal{E} \tag{8.21h}$$

$$z_{e,r}, v_{c,r} \in \{0,1\} \tag{8.21i}$$

$$s_c \in \mathbb{R}^+ \tag{8.21j}$$

Constraint (8.21e) ensures that events with locked rooms are assigned accordingly. Constraints (8.21f) and (8.21g) constraints variables $v_{c,r}$ and s_c properly, and thereby penalizes room stability. Constraint (8.21h) enforces that an event cannot be assigned a room if it not assigned to a timeslot, unless the event is locked to a specific room.

Notice that the room stability constraints (8.21f) and (8.21g) of Model (8.21) are not handled in any way in the Stage I model. Additional constraints which handle these constraints would theoretically improve the decomposition. It is however not trivial to model these constraints as a matching problem in a bipartite graph, so another approach may be required. This is a subject for future research. Apart from the room stability constraints, all other constraints are optimally handled by the decomposition, with the exception of the room allocation penalties, which are only partially integrated in the Stage I model.

8.6 Computational Results

For implementation purposes, Gurobi 5.0.1 was used as MIP solver on a machine with an Intel Core i7 930@2.80GHz CPU and 12GB of RAM, running Windows 8 64bit. Default parameter settings were used, and the interface was C# 4.5. The problem instances have been taken directly from the Lectio database, and are the same ones used in Sørensen and Stidsen (2013). These 100 real-world datasets provide a substantial ground for concluding on the numerical experiments. Note that 3 of these instances are available in the XHSTT format (Post et al. (2012a)) at `http://www.utwente.nl/ctit/hstt/`. We plan to make additional datasets available in this format in the future.

A time limit of 7200 seconds was imposed (6480 seconds for Stage I, and 720 seconds for Stage II), and Gurobi was allowed to use 8 threads. For the Stage I model, the initial solution given to Gurobi consists of assigning all events to the dummy-timeslot, except for those events locked to a specific timeslot. The initial solution for the Stage II model is analogous; Events are assigned to the dummy-room or the room which the event is locked too. A single run was used to establish results, as Gurobi has deterministic behavior.

Two other solution approaches are described for the same timetabling problem in Sørensen and Stidsen (2013). These are used in comparison with the algorithm of this paper, and are briefly described below:

- The 'usual' formulation using a three-index binary variable, denoted *3-index model* in the following. This is solved using Gurobi with standard settings, with a time limit of 7200 seconds. The objectives listed are the result of a single run.

- An *Adaptive Large Neighborhood Search* heuristic, denoted *ALNS* in the following. The reported objectives for this method is the average obtained over 10 runs, each run with a timelimit of 240 seconds. Hence, the comparison of objectives wrt. this heuristic is not 'fair', but it will be seen that even with this shorter timelimit, the ALNS in general performs best. This method is the one currently used by the customers of Lectio.

Furthermore, we test the described decomposition both with and without the room penalties added to the Stage 1 model (i.e. Model (8.20) with and without equations (8.20e), (8.20f) and (8.20g)). Thereby an empirical test of the effect of extending Stage I is performed. In the following, these two methods are denoted *TSD* and TSD^{RoomLB}, respectively. Notice that the results for TSD can also be found in the technical report Sørensen and Stidsen (2013).

Table 8.1 shows the obtained results. Table 8.2 summarizes some key numbers. Table 8.3 gives a summary for the three exact methods. A gap between an IP objective z and a lower bound LB is calculated by $100\frac{z-\text{LB}}{z}$.

Table 8.1: Computational results. For each dataset is shown the objective 'Obj' obtained by each method. For the IP-based approaches, also the best found lower bounds 'LB' are shown (i.e. for the 3-index model is shown the final value of the LP-relaxation used internally by Gurobi, and for TSD is shown the final value of the LP-relaxation of the Stage I model, as described in Section 8.4). If a solution is best overall, it is marked in **bold**. If a bound is best overall, it is marked with a '*'. For the two-stage approach of this paper is shown the runtime 'Time' and final gap 'Gap' found by Gurobi for both Stage I and Stage II. For Stage II, column 'Diff.' denotes the difference between the lower bound for room allocation of Stage I, and the actual matching obtained by Stage II (excluding room stability). Column '$\overline{\text{Gap}}$' denotes the best overall gap, i.e. the gap between the best available solution and the best available bound.

	Previous methods					$\text{TSD}^{\text{RoomLB}}$							
	ALNS	3-index model		TSD		Stage I		Stage II					
Dataset	Obj	Obj	LB	Obj	LB	Time	Gap	Time	Gap	Diff.	Obj	LB	$\overline{\text{Gap}}$
AalborTG2012	6317	6118	*5946	6018	5934	>6480	0.7	4	0.0	0	**6005**	5941	1.0
AarhusA2011	**10037**	58015	-	15872	*5986	>6480	66.6	154	0.0	160	18122	5985	40.4
AarhusA2012	**7971**	17096	5722	8947	*6005	>6480	49.4	108	0.0	48	11936	5962	24.7
Aars2009	**14900**	49504	-	20780	11874	>6480	47.9	14	0.0	0	24240	*12641	15.2
Aars2010	**16268**	81970	-	25057	13134	>6480	42.7	22	0.0	1	24692	*14151	13.0
Aars2011	**14256**	77967	-	30623	9709	>6480	68.9	10	0.0	3	33790	*10501	26.3
Aars2012	**10701**	55049	-	21206	7456	>6480	60.3	21	0.0	1	20274	*8044	24.8
Alssund2010	**9967**	52717	-	23173	6811	>6480	67.9	324	0.0	8	21455	*6876	31.0
Alssund2012	**29803**	108810	-	108810	-	>6480	-	6	0.0	0	108810	-	-
BagsvaG2010	3960	6777	3171	**3916**	3063	>6480	19.4	14	0.0	9	4051	*3227	17.6
BirkerG2011	**42063**	119600	-	119600	-	>6480	-	7	0.0	0	119600	-	-
BirkerG2012	19552	110180	-	19322	15662	>6480	0.9	>720	1.0	16	**18182**	*17709	2.6
BjerrG2009	**16877**	52639	-	35514	11094	>6480	55.2	11	0.0	0	27396	*12288	27.2
BjerrG2010	**4983**	12868	*3928	5788	3868	>6480	33.1	13	0.0	37	5977	3925	21.2
BjerrG2011	**6334**	13009	*4142	9302	4060	>6480	64.8	97	0.0	20	11676	4079	34.6
BjerrG2012	**8023**	17200	*5055	15265	5007	>6480	71.2	160	0.0	16	17404	4991	37.0
BroendG2012	2040	2005	*1881	1929	1859	1028	0.0	17	0.0	5	**1928**	1877	2.4
CPHWGym2010	**6775**	34415	-	19363	*3759	>6480	77.4	11	0.0	0	16589	3752	44.5
CPHWGym2011	**5679**	38232	-	16212	4095	>6480	72.7	10	0.0	0	15046	*4103	27.8
CPHWGym2012	**6762**	40945	-	15543	4205	>6480	75.5	19	0.0	1	17194	*4215	37.7
CPHWHG2012	**11077**	46625	8157	23088	*8338	>6480	64.1	16	0.0	0	23219	8326	24.7
CPHWHTX2010	**11342**	27174	9179	15943	8828	>6480	52.1	7	0.0	0	19314	*9259	18.4
CPHWHTX2011	20734	22466	20460	20708	18490	>6480	0.6	6	0.0	11	**20632**	*20470	0.8
CPHWHTX2012	**16256**	25998	14481	21392	13115	>6480	35.4	4	0.0	0	22481	*14531	10.6
DetFG2012	7560	8017	*7168	7265	7018	>6480	0.7	8	0.0	68	**7258**	7116	1.2
DetKG2010	**2947**	6058	1732	4006	*1821	>6480	55.6	3	0.0	4	4102	1814	38.2

Table 8.1 – continued from previous page

	Previous methods					TSD^{RoomLB}							
	ALNS	3-index model		TSD		Stage I		Stage II					
Dataset	Obj	Obj	LB	Obj	LB	Time	Gap	Time	Gap	Diff.	Obj	LB	$\overline{\text{Gap}}$
DetKG2011	**2820**	5594	1732	4366	1780	>6480	60.9	2	0.0	2	4577	*1781	36.8
EUCN2009	**3737**	7557	2911	4298	2856	>6480	40.3	4	0.0	0	5001	*2982	20.2
EUCN2010	3882	4231	3329	3463	3246	>6480	1.4	6	0.0	1	**3430**	*3375	1.6
EUCN2011	1468	1435	*1395	1430	1384	>6480	2.2	1	0.0	2	**1426**	1384	2.2
EUCN2012	**3289**	9430	2327	5059	*2363	>6480	60.2	4	0.0	0	5913	2359	28.2
EUCNHG2010	1505	1476	1371	1421	1368	>6480	2.1	2	0.0	0	**1408**	*1378	2.1
EUCS2012	3714	4689	3576	3783	3347	>6480	3.0	3	0.0	0	**3695**	*3584	3.0
FaaborgG2008	**68124**	125330	-	125330	-	>6480	-	14	0.0	0	125330	-	-
FalkonG2009	**10449**	88890	-	88890	-	>6480	-	5	0.0	0	88890	-	-
FalkonG2011	**8584**	76170	-	16543	*5183	>6480	75.9	>720	0.0	48	20758	4953	39.6
FalkonG2012	**10143**	100190	-	16666	*6105	>6480	58.6	>720	0.1	121	14908	6050	39.8
GUAasia2010	6527	6579	6354	6461	6035	26	0.0	>720	0.1	5	**6422**	*6374	0.7
GUQaqor2011	**6674**	19623	4537	10005	*4554	>6480	59.9	3	0.0	18	11396	4542	31.8
GUQaqor2012	**5733**	11488	4314	7619	4294	>6480	55.1	10	0.0	0	9650	*4324	24.6
HadersK2011	**7128**	51190	-	14229	*3909	>6480	76.2	>720	0.0	43	16494	3888	45.2
HasserG2010	**11963**	96790	-	96790	-	>6480	-	6	0.0	0	96790	-	-
HasserG2011	**16061**	99840	-	99840	-	>6480	-	6	0.0	0	99840	-	-
HasserG2012†	**18338**	112160	-	112034	-	>6480	-	7	0.0	0	112160	-	-
HerningG2010	**37**	**37**	*37	**37**	35	0	0.0	1	0.0	0	**37**	35	0.0
HerningG2011	**15091**	163785	-	23117	*9829	>6480	61.8	108	0.0	169	26410	9746	34.9
HerningG2012	**13147**	185433	-	14952	9763	>6480	48.4	>720	0.1	262	19834	*9817	25.3
HoejeTaG2008	2958	6292	2253	**2707**	2563	>6480	6.4	3	0.0	0	2775	*2587	4.4
HoejeTaG2009	**9157**	45260	-	26066	*5773	>6480	79.7	105	0.0	3	27779	5628	37.0
HoejeTaG2010	**9862**	45095	-	25678	*6188	>6480	78.1	106	0.0	5	27886	6116	37.3
HoejeTaG2011	**10158**	51050	-	32630	*6726	>6480	78.2	66	0.0	3	30327	6601	33.8
HoejeTaG2012	**12502**	72455	7592	18627	7845	>6480	79.8	9	0.0	3	39326	*7952	36.4
HorsenS2009	3111	**3100**	*3100	**3100**	2865	1	0.0	4	0.0	13	**3100**	3059	0.0
HorsenS2012	**10056**	86090	-	86090	-	>6480	-	3	0.0	0	86090	-	-
Johann2012	**23001**	92575	-	27781	18456	>6480	33.5	233	0.0	6	29491	*19590	14.8
KalundG2011	**38479**	126150	-	126150	-	>6480	-	9	0.0	0	126150	-	-
KalundG2012	**26768**	123010	-	123010	-	>6480	-	11	0.0	0	123010	-	-
KalundHG2010	**5631**	12103	4540	6351	4551	>6480	29.7	6	0.0	0	6605	*4642	17.6
KoebenPG2012	888	1872	637	**874**	642	>6480	37.9	1	0.0	2	1052	*645	26.2
KoegeH2012	**11418**	108347	-	20150	9096	>6480	53.7	12	0.0	0	20390	*9440	17.3
KongshoG2010	**4296**	8889	2411	7954	*2488	>6480	65.9	30	0.0	0	7208	2451	42.1
MariageG2009	**8013**	54030	-	20138	5118	>6480	69.7	>720	0.0	18	17506	*5286	34.0
MorsoeG2012	**5651**	42762	-	10241	3854	>6480	66.0	23	0.0	6	11674	*3947	30.2
NaerumG2008	**24104**	118370	-	117894	-	>6480	-	7	0.0	0	117894	-	-
NaerumG2009	7667	100450	-	6681	*5114	>6480	0.3	>720	6.2	0	**5466**	5113	6.4
NielsSG2011	**4953**	10464	3323	6132	*3412	>6480	37.6	9	0.0	0	5397	3367	31.1
NielsSG2012	**6952**	12747	5722	8003	*5738	>6480	37.6	14	0.0	4	9192	5724	17.5
NordfynG2012	5160	8201	*4152	**4890**	4048	>6480	23.3	35	0.0	59	5510	4107	15.1
NyborgG2011	**13944**	94059	-	31809	*6129	>6480	-	7	0.0	4	85816	-	56.0
OdderCfU2010	**18219**	59540	-	40032	12188	>6480	66.9	66	0.0	2	38875	*12865	29.4
OdderG2009	**9308**	59851	-	57586	-	>6480	78.1	67	0.0	67	24686	*5361	42.4
OdderG2012	**12307**	17402	9602	14888	8878	>6480	64.2	4	0.0	57	27199	*9688	21.3
OrdrupG2010	13663	75700	-	**12936**	10665	>6480	39.5	10	0.0	0	18101	*10810	16.4
OrdrupG2011	**21612**	116400	-	31329	16904	>6480	38.7	305	0.0	8	28884	*17692	18.1
RibeK2011	**21679**	61945	-	43175	16209	>6480	53.8	229	0.0	5	39107	*18055	16.7
RysenG2010	**39971**	110690	-	110690	-	>6480	-	6	0.0	0	110690	-	-
RysenG2011	**22260**	100313	-	25989	17756	>6480	71.4	9	0.0	5	68927	*19725	11.4
RysenG2012	**19841**	110111	-	22156	15115	>6480	71.7	14	0.0	15	59124	*16708	15.8
SanktAG2012	4207	4624	3415	3911	3376	>6480	0.7	>720	0.5	39	**3721**	*3538	4.9
SkanderG2010	7209	7708	6051	6875	5712	>6480	0.6	>720	0.5	72	**6485**	*6238	3.8
SkanderG2011	**22525**	88470	-	88470	-	>6480	-	5	0.0	0	88470	-	-
SkanderG2012	**20138**	98487	-	95319	-	>6480	-	7	0.0	3	95319	-	-

Table 8.1 – continued from previous page

	Previous methods					TSD^{RoomLB}							
	ALNS	3-index model		TSD		Stage I		Stage II					
Dataset	Obj	Obj	LB	Obj	LB	Time	Gap	Time	Gap	Diff.	Obj	LB	Gap
SkiveG2010	**43120**	194740	-	194740	-	>6480	-	526	0.0	0	194740	-	-
SlagelG2012†	**32167**	162960	-	162765	-	>6480	-	417	0.0	0	162960	-	-
SoendS2011	**11776**	83560	-	83560	-	>6480	-	131	0.0	0	83560	-	-
SoendS2012	**8420**	17778	*6838	11915	6647	>6480	72.5	8	0.0	4	24668	6739	18.8
StruerS2012	**73361**	-	-	207488	-	>6480	-	700	0.0	0	211960	-	-
VardeG2012	**10777**	20933	*5921	20622	5720	>6480	72.1	12	0.0	2	20496	5668	45.1
VejenG2009	**11264**	69450	-	69450	-	>6480	73.9	>720	0.0	7	27954	*7290	35.3
Vejlefjo2011	**13514**	52035	-	18043	8511	>6480	60.0	456	0.0	3	22066	*8805	34.8
VestfynG2009	5973	11606	4176	5999	4137	>6480	14.5	553	0.0	18	**5032**	*4211	16.3
VestfynG2010	6761	16895	*4308	5974	4225	>6480	16.3	65	0.0	21	**5239**	4290	17.8
VestfynG2011	7013	13624	5110	6657	4925	>6480	19.6	38	0.0	24	**6522**	*5159	20.9
VestfynG2012	5244	11095	4279	**5212**	4210	>6480	17.5	149	0.0	21	5319	*4315	17.2
ViborgK2011	**14923**	99170	-	99170	-	>6480	-	6	0.0	0	99170	-	-
ViborgTG2009	**10216**	19891	8695	12077	8356	>6480	34.7	45	0.0	3	13387	*8740	14.4
ViborgTG2010	**4932**	12727	4130	10226	3990	>6480	60.9	11	0.0	19	10665	*4146	15.9
ViborgTG2011	**7478**	16433	6716	9808	6204	>6480	38.8	12	0.0	13	11088	*6772	9.4
VirumG2012	**27738**	140883	-	32183	17770	>6480	75.3	16	0.0	9	79111	*19486	29.7
VordingbG2009	**8568**	17025	5457	9905	5243	>6480	33.6	10	0.0	167	8972	*5787	32.5
Avg.							44.3		0.1	17.9			22.3

† An artificial bound on the amount of Halls' inequalities was enforced for tractability, see Section 8.4.3.

Table 8.2: Results summary. Note that rows 'Best solution' and 'Best bound' also counts draws.

	ALNS	3-index model	TSD	TSD^{RoomLB}
Solution found	100	99	100	100
Best solution	77	2	8	18
Bound found	-	46	79	80
Best bound	-	13	19	49

Table 8.3: Comparison of the amount of best found solutions for the exact methods. Draws are also counted.

	3-index model	TSD	TSD^{RoomLB}
Best solution	16	66	52

A number of conclusions can be drawn from the numbers:

- For 97 instances, the solution obtained by TSD^{RoomLB} is at least as good as the solution obtained by the 3-index model, and for most instances significantly better.
- TSD^{RoomLB} is generally the best method for generating bounds, finding the best bound on 49 instances overall.
- TSD^{RoomLB} was capable of finding a lower bound for 80 instances. This means that for 20 instances, Gurobi was unable to solve the LP-relaxation of the root node of the Stage I model within the timelimit. Table 8.4 shows statistics for these instances (the presolved models). It is seen that the problems does not contain coefficients of huge magnitude in

neither the objective, system matrix, or rhs. Hence the issue seems related to the relatively big number of constraints and variables. In average, these instances have more than 100000 constraints and variables, hence we think its fair to consider them as large-scale. Note that if the root-LP was not solved for an instance, the reported solution replicates the initial solution provided by us to Gurobi (Gurobi apparently starts its solution process by verifying the feasibility of the MIP Start attributes).

- $\text{TSD}^{\text{RoomLB}}$ produces the best solution for 18 instances overall, while TSD produces the best solution on 8 instances. The ALNS heuristic is best on 77 instances, and is currently the best algorithm for this problem (keep in mind the ALNS algorithm was allowed significantly less CPU time).
- Comparing the exact methods (Table 8.3), it is seen that it is generally not profitable to use the extended Stage I model if the goal is to obtain good solutions. Both variants of the decomposition finds more best solutions than the pure 3-index model.
- The Stage I model of $\text{TSD}^{\text{RoomLB}}$ is a challenge for Gurobi, with an average gap of 44.3% over all instances where a LB was found. Only 4 instances are solved to optimality, and these are among the smallest instances (see Sørensen and Stidsen (2013) for instance statistics).
- The Stage II model of $\text{TSD}^{\text{RoomLB}}$ is in general easy to solve. The average gap for this model over all problem instances is 0.1%, and 89 instances are solved to optimality. This means that future research can focus on solving the Stage I model.
- The difference between the lower bound on room allocation and the actual allocation of rooms (column 'Diff.') is low, compared to the magnitude of objectives in general. This means that only a small increase in solution quality can be gained by improving the bound on room allocation.

Table 8.4: Statistics of the Stage I models (after presolve) where Gurobi was unable to solve the LP-relaxation of the root-node within the timelimit (i.e. for those instances where the $\text{TSD}^{\text{RoomLB}}$ was unable to provide a lower bound). Column 'Cons.' shows the number of constraints and 'Non-zeros' shows the amount of non-zeros in the model. 'Variables' shows the amount of continuous, integer and binary variables. 'Obj. coef.', 'Model coef.', and 'RHS coef' shows the smallest and largest coefficient in the objective function, system matrix and right-hand side, respectively.

			Variables			Obj. coef.		Model coef.		RHS coef.	
	Cons.	Non-zeros	Cont.	Integer	Binary	Min.	Max.	Min.	Max.	Min.	Max.
Min.	73293	881433	24089	52518	41121	1	84	1	4	0	6
Max.	167978	4200206	52015	252514	246874	1	1120	1	16	1	103
Avg.	118396	1967455	36918	101460	88321	1	264	1	7	1	32

As an extension to the decomposition, one could use the ALNS heuristic to provide a starting solution. Since the ALNS heuristic is able to produce a fairly good solution quickly, this would most likely lead to improved performance.

As a loose remark, we mention that Burke et al. (2010) formulates an IP of the *Udine Course Timetabling Problem* (used in the International Timetabling Competition 2007), using a three-indexed binary variable, and reports that CPLEX 11 uses up to 6400 seconds when solving

the root LP (using Dual Simplex, which is also used by Gurobi as default). Their model is quite similar in structure to ours, so possibly this class of IP formulations contain undesirable properties in the eyes of general-purpose MIP solvers.

8.7 Conclusion

A Two-Stage Decomposition for a real-world high school timetabling problem has been shown. This splits the Integer Programming model into two smaller models, which reduces the number of variables significantly. Computational results show that this approach is way more effective than solving the usual original IP with a 3-index binary variable, in terms of both the obtained solutions and the obtained bounds. This constitutes the TSD as the best exact method for solving this particular timetabling problem. However, the integration of the lower bound on room allocation in the Stage I model has bad influence on the quality of solutions, but makes the decomposition capable of achieving better lower bounds. Nevertheless this extension of the Stage I model represents interesting theory which can likely be used in the context of decomposing other timetabling problems.

For other types of (timetabling) problems, this type of decomposition might be a way of enhancing computational times. However, a special structure is required for applying the decomposition, which limits the set of applicable problems. On the other hand, the advantage gained by reducing the number of variables should not be underestimated, and we encourage researchers to attempt this type of decomposition if possible.

Approximating the room allocation penalties in the Stage I model is an interesting approach, and also sets a possible agenda for future work; 1) Can a better approximation (or even the exact value) be found for the minimum weight maximum matching problem representing room penalties? 2) Can the room stability penalties be incorporated in the Stage I model? However, the most important issue for future research is a more efficient way of solving the Stage I model. This is the bottleneck of the TSD for this particular problem.

Bibliography

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

M. A. Badri. A two-stage multiobjective scheduling model for [faculty-course-time] assignments. *European Journal of Operational Research*, 94(1):16 – 28, 1996. ISSN 0377-2217.

E. Balas and W. Pulleyblank. The perfectly matchable subgraph polytope of a bipartite graph. *Networks*, 13(4):495–516, 1983. ISSN 1097-0037.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

T. Birbas, S. Daskalaki, and E. Housos. Timetabling for greek high schools. *Journal of the Operational Research Society*, 48:1191–1200(10), 1997.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

E. Burke and J. P. Newall. A multistage evolutionary algorithm for the timetable problem. *Evolutionary Computation, IEEE Transactions on*, 3(1):63–74, 1999. ISSN 1089-778X.

E. Burke, J. Marecek, A. Parkes, and H. Rudová. Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 37(3):582–597, 2010.

M. Carter. A decomposition algorithm for practical timetabling problems. Technical Report 83-06, Department of Industrial Engineering, University of Toronto, 1983.

S. Daskalaki and T. Birbas. Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160(1):106 – 120, 2005. ISSN 0377-2217.

S. Daskalaki, T. Birbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.

D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2): 151 – 162, 1985. ISSN 0377-2217.

M. Dimopoulou and P. Miliotis. Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, 130(1):202 – 213, 2001. ISSN 0377-2217.

J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

E. Egerváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38:16–28, 1931.

C. C. Gotlieb. The construction of class-teacher timetables. In C. M. Popplewell, editor, *IFIP Congress*, volume 62, pages 73–77, North-Holland Pub. Co, 1962.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

N. L. Lawrie. An integer linear programming model of a school timetabling problem. *The Computer Journal*, 12(4):307–316, 1969.

L. Lovász and M. D. Plummer. *Matching Theory*. AMS Chelsea Publishing, 2009.

S. MirHassani. A computational approach to enhancing course timetabling with integer programming. *Applied Mathematics and Computation*, 175(1):814 – 822, 2006. ISSN 0096-3003.

H. Mittelman. Benchmarks for optimization software. `http://plato.asu.edu/bench.html` [Accessed 20/8-2013], Aug. 2013.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3): 230–238, 2003.

N. Pillay. A survey of school timetabling research. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

A. Qualizza and P. Serafini. A column generation scheme for faculty timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 161–173. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24 of *Algorithms and Combinatorics*. Springer, 2003.

M. Sørensen and T. Stidsen. Comparing solution approaches for a complete model of high school timetabling. Technical Report 5.2013, DTU Management Engineering, Technical University of Denmark, March 2013.

Chapter 9 Paper D

Decomposing the Generalized High School Timetabling Problem

Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract In the scientific literature, high school timetabling problems are generally solved by heuristics, and exact solution methods are not widely used. This paper presents a *Two-Stage Decomposition* (TSD) of an Mixed-Integer Programming (MIP) model for the *Generalized High School Timetabling Problem*, which is based on the publicly available XHSTT format. TSD is a recent method which has shown good results for other timetabling problems with similar structure, and it has a number of advantages over the original MIP model, most notably that the total amount of variables is significantly reduced. The method can be applied to all types of XHSTT instances, however in this paper it is tested on the 12 instances where the differences compared to the original MIP model are most notable. Whether or not the method is exact is dependent on specific characteristics of the dataset in question, and we provide a list of necessary requirements. Investigations of these 12 instances show that for 6 of these the decomposition is exact. For 9 of the instances, the decomposition is capable of providing lower bounds (the structure of the remaining 3 instances prevent the generation of lower bounds). Computational experiments on these 12 instances show that the decomposition yields significantly smaller models. However, the quality of the obtained solutions and lower bounds are not as high as expected compared to the original MIP model.

9.1 Introduction

The *High School Timetabling Problem* (HSTP) is a widely known optimization problem which have attracted researchers for many decades. Most of the work on this problem has been performed on specialized cases, often only applying to few educational institutions. The introduction of the *XHSTT format* (Post et al. (2012)) aims at bridging this gap by being widely applicable and providing many instances of HSTP of different origins and with different characteristics. The format was used in the *International Timetabling Competition 2011* (ITC2011), and its popularity in the community is rising. Kristiansen et al. (2013) introduces a *Mixed-Integer Linear Programming* (MIP) model for the XHSTT format. This MIP model was capable of solving some instances to optimality, and providing lower bounds on optimum for several other instances. Based on this MIP model, we present a *Two-Stage Decomposition* (TSD) approach.

The idea of TSD is to split the MIP model into two smaller MIP models, while maintaining optimality (as far as possible) of the original model. TSD was first introduced in Lach and Lübbecke (2008, 2012), where it was applied to the Curriculum-based University Timetabling Problem. This problem was considered in one of the tracks of the *International Timetabling Competition 2007* (ITC2007) (McCollum et al., 2010). Sørensen and Dahms (2014) applied TSD to another timetabling problem, the Danish case of high school timetabling, and suggested extensions of the decomposition to handle additional details of the optimization problem. The application of TSD is at present closely tied to the specific optimization problem, so a well-defined description of the method on an abstract level cannot be given. However, given an optimization problem for which TSD is applicable, the decomposition can be easily described (see Section 9.4). With the contributions of this paper, TSD has shown to be capable of handling instances from both ITC2007 and ITC2011, and is a promising branch of exact methods for timetabling problems. We see no reason that TSD could also be applied to other types of problems with similar structure, for instance *Nurse Rostering* (Santos et al., 2012).

The paper is structured as follows: Section 9.2 presents related literature. Section 9.3 describes the generalized HSTP in detail, including basic details of a MIP model. Section 9.4 gives details on the TSD for the problem. Section 9.5 outlines the solution method. Section 9.6 presents results from the computational experiments. Section 9.7 concludes on our findings.

9.2 Related Literature

The amount of attention given to TSD is still small within the timetabling community, so not much literature exists. However, some extensions of Lach and Lübbecke (2008, 2012) have been proposed, described in the following. Cacchiani et al. (2013) propose a new method for computing lower bounds, by splitting the problem based on the objective function. Hao and Benlic (2011) also develops lower bounds for the ITC2007 Track 3 instances, using a partition-based approach with an embedded Tabu Search. Both of these methods could in principle also be applied to the XHSTT case on the basis of the TSD described in this paper. Burke et al. (2010) use a multiphase approach for the ITC2007 instances where partial solutions are produced, build by subproblems of similar difficulty. A lower bound on a subproblem is also a lower bound on the overall problem. Daskalaki and Birbas (2005) do not consider the ITC2007 instances, but uses a two-stage approach, where the first stage is a relaxed version of the original problem. The second stage then solves small parts of the problem and established local optima.

In terms of XHSTT, Kristiansen et al. (2013) propose the first (and currently only) exact method, based on a MIP model. The presented TSD is based on this MIP. Heuristic approaches to XHSTT are mainly those of ITC2011. The finalists of the competition were Fonseca et al. (2012); Kheiri et al. (2012); Romrös and Homberger (2012); Sørensen et al. (2012). An obvious advantage of the exact methods is that they allow lower bounds to be found, which yields fair evaluation criteria for solution methods. Some recent contributions to heuristic methods for XHSTT are Shambour et al. (2013) which apply a hybrid of a Evolutionary Algorithm and Simulated Annealing, Pimmer and Raidl (2013) which apply a heuristic that fill one timeslot at a time, and Ter Braak (2012) which applies hyper-heuristics.

9.3 Problem Description - The XHSTT format

An instance of XHSTT contains a set of *events* $\mathcal{E}$, a set of *times* $\mathcal{T}$, and a set of *resources* $\mathcal{R}$. Each event has a duration $D_e \in \mathbb{N}^+$, and can be split into parts which are denoted *sub-events*. The duration denotes the amount of times which the event spans. The entire set of sub-events

is denoted $\mathcal{SE}$, and by $se \in e$ we denote that sub-event se is a part of event e. Each sub-event $se \in \mathcal{SE}$ has a duration $D_{se} \in \mathbb{N}^+$, and it applies that $D_{se} \leq D_e$.

Furthermore an event e contains a set of *event resources*, which we each denote by $er \in e$. An event resource describes a requirement of a resource, either one that is already known (a *preassigned* resource), or a resource of a certain type (a *free* resource). We denote by $r \in er$ that resource r can be assigned to event resource er. The parameter $PA_{er} \in \{0, 1\}$ takes value 1 if event resource er is preassigned, and 0 otherwise. Let $|er|_{se}$ denote the amount of event resources of sub-event $se \in \mathcal{SE}$.

The total duration of the *active* sub-events of event $e \in \mathcal{E}$ in a solution cannot exceed D_e. A sub-event is active if it assigned a time and/or has an event resource with a resource assigned, and this resource is not preassigned.

An XHSTT instance also contains a set of constraints $\mathcal{C}$, and each constraint $c \in \mathcal{C}$ is of a particular type, and applies to certain events or resources. A constraint penalizes certain characteristics of a solution to the instance. For instance, an *Assign Times* constraint penalizes events which are not assigned to a time.

Typically, a solution is desired which assigns events to times and resources. WLOG, we let the variable $x_{se,t,er,r} \in \{0, 1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is assigned to starting time $t \in \mathcal{T}$ and resource $r \in er$ is assigned to event-resource $er \in se$, and 0 otherwise. Variable $y_{se,t}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is assigned time $t \in \mathcal{T}$ as starting time, and 0 otherwise. Variable $w_{se,er,r}$ takes value 1 if event resource $er \in se$ of sub-event $se \in \mathcal{SE}$ is assigned resource $r \in er$, and 0 otherwise. Variable u_{se} takes value 1 if sub-event $se \in \mathcal{SE}$ is active, and 0 otherwise.

Model (9.1) shows a basic MIP model of XHSTT. Constraint (9.1b) ensures that each sub-event and corresponding event resource is assigned one time and one resource. Constraint (9.1c) links variables $x_{se,t,er,r}$ and $y_{se,t}$, by ensuring that the amount of resources assigned to a sub-event $se \in \mathcal{SE}$ at a particular time $t \in \mathcal{T}$ corresponds to $|er|_{se} y_{se,t}$. Together with constraint (9.1b) this ensures that a sub-event is only assigned to one starting time. Constraint (9.1d) links variables $x_{se,t,er,r}$ and $w_{se,er,r}$. To ensure that a feasible solution exists, the set of times and the set of resources are both extended with a dummy-element, i.e. $\mathcal{T} = \{\mathcal{T} \cup t_D\}$ and $\mathcal{R} = \{\mathcal{R} \cup r_D\}$. These dummy elements are necessary due to the equality sign of constraints eqs. (9.1b) to (9.1d), and correspond to the case where a sub-events or a event-resource is *not* assigned a time or a resource, respectively.

Constraint (9.1e) makes sure that if a sub-event is assigned a time, it is also marked as active. Likewise, Constraint (9.1f) ensures that if a sub-event is assigned a non-preassigned resource it is marked as active. Constraint (9.1g) ensures that if a sub-event is not assigned a time or a non-preassigned resource, it cannot be marked as active. Constraint (9.1h) makes sure that the sum of the duration of the active sub-events of an event sum to the total duration of the event.

Furthermore, all constraints $\mathcal{C}$ of the XHSTT format (see Post (2013)) are included in Constraint (9.1i), slightly abusing notation. For each constraint $c \in \mathcal{C}$ the slack variable $s_{c,p} \in \mathbb{N}_0$ is introduced, which denote the penalty incurred for point-of-application p of constraint c by the setting of values in $x_{se,t,er,r}$. This should be understood in the way that the values taken by $x_{se,t,er,r}$ inevitably yields specific penalty values for all XHSTT constraints, since this is the basic variable of the MIP model. A MIP formulation of the XHSTT constraints (i.e. formulated in terms of $x_{se,t,er,r}$) can be found in Kristiansen et al. (2013).

By definition of the objective of XHSTT, with each constraint $c \in \mathcal{C}$ is associated a specific CostFunction (which can be either one of the following: *Sum*, *SumSquare*, *SquareSum*, *SumStep*, *StepSum*) and a weight $w_c \in \mathbb{N}_0$. Given the penalty value $s_{c,p}$, the contribution to the objective of constraint c is $w_c \cdot \text{CostFunction}(s_{c,p})$. This constitutes the objective of the MIP model. Furthermore, each constraint is either a *hard constraint* or a *soft constraint*. The violation of these constraints is denoted *hard cost* and *soft cost*, respectively. A solution which respects all

hard constraints, meaning that the hard cost is 0, is said to be feasible. Solutions to a XHSTT instances are ranked by the hard cost first, and the soft cost second. How we deal with this structure of the objective is described in Section 9.6.

XHSTT MIP model (9.1)

$$\min \quad \sum_{c \in \mathcal{C}} w_c \cdot \text{CostFunction}(s_{c,p}) \tag{9.1a}$$

s.t.

$$\sum_{t \in \mathcal{T}, r \in er} x_{se,t,er,r} = 1 \qquad \forall se \in \mathcal{SE}, er \in se \tag{9.1b}$$

$$\sum_{er \in se, r \in er} x_{se,t,er,r} = |er|_{se} \cdot y_{se,t} \quad \forall se \in \mathcal{SE}, t \in \mathcal{T} \tag{9.1c}$$

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \qquad \forall se \in \mathcal{SE}, er \in se, r \in er \tag{9.1d}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} \leq u_{se} \qquad \forall se \in \mathcal{SE} \tag{9.1e}$$

$$\sum_{r \in er \setminus r_D} w_{se,er,r} \leq u_{se} \qquad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{9.1f}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} + \sum_{\substack{r \in er \setminus r_D \\ er \in se, PA_{er} = 0}} w_{se,er,r} \geq u_{se} \qquad \forall se \in \mathcal{SE} \tag{9.1g}$$

$$\sum_{se \in e} D_{se} \cdot u_{se} = D_e \qquad \forall e \in \mathcal{E} \tag{9.1h}$$

$$(x_{se,t,er,r}, s_{c,p}) \in \mathcal{C} \tag{9.1i}$$

$$x_{se,t,er,r}, y_{se,t}, w_{se,er,r} \in \{0,1\} \tag{9.1j}$$

$$s_{c,p} \in \mathbb{N}_0 \tag{9.1k}$$

9.4 Two-Stage Decomposition

The decomposition is performed as follows. Model (9.1) is split into two smaller MIP models, denoted Stage I and Stage II, respectively. Loosely speaking, the aim of Stage I is to assign sub-events to times, and the aim of Stage II is to assign sub-events to resources, subject to the assignment to times performed in Stage I. The key advantage of this approach is that variable $x_{se,t,er,r}$ is no longer necessary, which significantly reduces the total amount of variables. Specifically, the basic variables of the two stages are $y_{se,t}$ and $w_{se,er,r}$, respectively. The decomposition progresses as follows: Stage I is solved, obtaining solution $y^*_{se,t}$. This solution is not necessarily the optimal one. $y^*_{se,t}$ is given as input to Stage II, where it is used as a parameter. Stage II is solved, obtaining a solution $w^*_{se,er,r}$. A solution to the original model is found by calculating $x_{se,t,er,r} = y^*_{se,t} w^*_{se,er,r}$ $\forall se \in \mathcal{SE}, t \in \mathcal{T}, er \in se, r \in er$. Clearly, Stage I is a relaxation of the original model, and the lower bound obtained by solving Stage I is therefore a lower bound on the original model. To summarize, Figure 9.1 shows the outline of TSD.

A discussion is needed on how the set of constraints $\mathcal{C}$ (Constraint (9.1i)) should be handled in this decomposition. Obviously, constraints which only base their evaluation on the assignment of events to times can be optimally handled in Stage I. Furthermore, constraints which deal only with preassigned resources can also be handled optimally in Stage I, as full information is

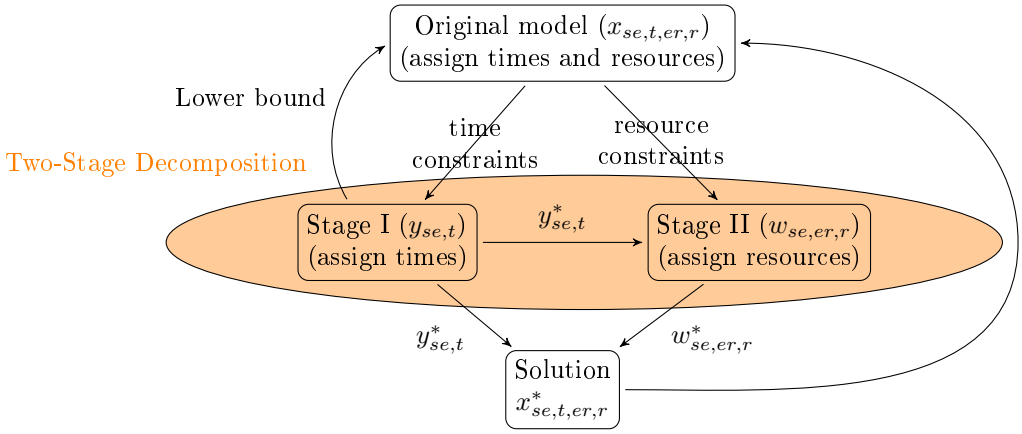


Figure 9.1: Outline of Two-Stage Decomposition (Sørensen and Dahms, 2014).

available. This is done by introducing the parameter $\overline{w}_{se,er,r}$ in Stage I, which can be used to derive $x_{se,t,er,r}$ in combination with $y_{se,t}$. The issue that remains is therefore constraints which deal with free resources, known as *free resource constraints* in the following. This is discussed in Section 9.4.1.

9.4.1 Handling of free resources

By construction, Stage II is solved subject to the solution of Stage I. This is unfortunate as optimality of the original model might be lost in the process, as Stage I is not aware of the objective of Stage II. A key point of TSD is therefore to consider an extension of Stage I which also consider the objective of Stage II. In our case, we aim at making Stage I perform an assignment of times which allows the optimal assignment of resources in the subsequent stage. If such an extension is found, the TSD is an exact method.

Denote by A the set of all eligible combinations of sub-events and event-resources, i.e. $A = \{(se, er) \mid se \in \mathcal{SE}, er \in se\}$. Consider the graph G, which contains a vertex for every element in A and a vertex for every resource $r \in \mathcal{R}$. All edges of the graph E connects an element of A and a resource vertex, and denotes that this is a feasible assignment of the resource. Clearly this is a *bipartite* graph, and we denote it by $G = (A \cup \mathcal{R}, E)$. An assignment of resources is clearly a matching in this graph, i.e. for each vertex in A select one adjacent edge such that no vertex of $\mathcal{R}$ is adjacent to more than one vertex. Thereby the matching problem of this graph resembles Stage II in its most basic form. Exploiting properties of this graph allows us to enhance the TSD.

Denote by $\Gamma(S)$ the neighbors of $S \subseteq A$ in graph G, i.e. $\Gamma(S) = \{\, r \mid (a, r) \in E, a \in S, r \in \mathcal{R} \,\}$ The famous theorem of Hall state the following:

Theorem 8 (Hall's Theorem)**.** *The bipartite graph $G = (A \cup \mathcal{R}, E)$ has a matching of all vertices A into $\mathcal{R}$ if and only if $|\Gamma(S)| \geq |S| \;\; \forall S \subseteq A$.*

Furthermore, let the *deficiency* of vertex set $S \subseteq A$ be defined as $\text{def}(S) = |S| - |\Gamma(S)|$. Let the deficiency of G be defined as $\text{def}(G) = \max_{S \subseteq A} \text{def}(S)$. The *matching number* of G is defined as $\nu(G) = |A| - \text{def}(G)$, and denotes the number of elements of A which are matched in a maximum matching.

Recall that in Stage II, the times of all sub-events are fixed by $y^*_{se,t}$. Consider the graph $G_t = (A_t \cup \mathcal{R}, E)$, where A_t denotes the element of A where the sub-event is assigned to time

$t \in \mathcal{T}$. Thereby G_t is a *subgraph* of G. Let $\Gamma_t(S)$ denote the neighbors of $S \subseteq A_t$ in graph G_t. By Theorem 8, a matching of all elements in A_t into $\mathcal{R}$ exists iff $\Gamma_t(S) \geq |S| \ \forall S \subseteq A_t$. Let $T^{\text{start}}_{se,t} \subseteq \mathcal{T}$ be set the of times which are used by sub-event $se \in \mathcal{SE}$ if its starting time is $t \in \mathcal{T}$ (this parameter is derived from the duration of the sub-event). Consider the following two observations:

- Denote by $se \in S \subseteq A$ all elements of A which contain sub-event $se \in \mathcal{SE}$. $|S|$ can be substituted by $\sum_{se \in S, t' \in T^{\text{start}}_{se,t}} y_{se,t'}$, i.e. the amount of sub-events which are assigned a starting-time which results in them lying in time $t \in \mathcal{T}$. An auxiliary variable could be introduced which determines all times which a sub-event lies in, which would reduce the amount of non-zeros in the model. This has not been done as it would increase the amount of variables in the model.
- Assuming that we seek to minimize the deficiency, the deficiency of graph G_t can be derived by applying the constraints (here $\text{def}(G_t)$ takes the role of a variable),

$$\sum_{se \in S, t' \in T^{\text{start}}_{se,t}} y_{se,t'} - \text{def}(G_t) \leq |\Gamma_t(S)| \ \forall S \subseteq A \tag{9.2}$$

How we deal with the exponential amount of constraints of eq. (9.2) is discussed in Section 9.4.3.

To summarize, we state the following: A necessary and sufficient condition has been found for the assignment of resources to all event resources. Eq. (9.2) state this condition for a specific time $t \in \mathcal{T}$, incorporating a slack variable $\text{def}(G_t)$ which determines the number of (sub-event, event resource) elements *not* assigned a resource. The condition is stated in terms of variables of Stage I only, and thereby Stage I can be made aware of the resource assignment performed in Stage II. If the slack variable $\text{def}(G_t)$ has value 0, all event-resources can be assigned a resource.

What we can determine using eq. (9.2) is the *amount* of (sub-event, event resource) elements not assigned a resource for a particular time $t \in \mathcal{T}$, contrary to *which* specific elements this will ultimately turn out to be. This allows some of the XHSTT constraints w.r.t. free resources to be evaluated optimally or partially. In the following each constraint is discussed. The XHSTT constraints *Assign Time*, *Split Events*, *Distribute Split Events*, *Prefer Times*, *Spread Events*, *Link Events*, and *Order Events* only considers time assignments and can be optimally handled in Stage I. Constraints *Limit Idle Times*, *Cluster Busy Times*, *Limit Busy Times* and *Limit Workload* involve the evaluation of characteristics for a specific resource across multiple times, which cannot be done using conditions similar to eq. (9.2). Hence these constraints can not be taken into account by the decomposition if they apply to free resources, however in practice this has little importance, as discussed in Section 9.4.2.

The remaining constraints are *Assign Resource*, *Prefer Resource*, *Avoid Split Assignments*, *Avoid Clashes* and *Avoid Unavailable Times*, denoted by the set $\mathcal{C}^h$ in the following. These constraints are modeled by altering the set of edges and their corresponding weights in the bipartite graphs, however this is not sufficient for all constraints $\mathcal{C}^h$ (as described below). Consider the bipartite graph $G = (A \cup \mathcal{R}, E)$ with edge weights $w_e \in \mathbb{N}_0 \ \forall e \in E$. As a starting point we add an edge for every element in A which have a prefixed resource, and for those elements not having a prefixed elements an edge to every possible resource of the appropriate *type* is added. In the following details are provided on how to model each constraint $c \in \mathcal{C}^h$.

- *Assign Resource*: This constraint penalizes event resources which are not assigned a resource. It applies to a certain set of events $E^{\text{assignresource}}_c \subseteq \mathcal{E}$, and the entire set of resources $\mathcal{R}$. For each $e \in E^{\text{assignresource}}_c$ a corresponding dummy-vertex is added to G, along with

an edge having the weight w_c for the *Assign Resource* constraint $c \in \mathcal{C}$. This represents the option to *not* assign a resource to the event, and is penalized accordingly.

- *Prefer Resources*: This constraint penalizes event resources which are assigned resources that are not among the preferred resources. This is modeled in the graph by adding the weight of the *Prefer Resource* constraint $c \in \mathcal{C}$ to edges which models an assignment of a resource which is not preferred.
- *Avoid Split Assignments*: This constraint tries to avoid the assignment of different resources to the same event resource of sub-events for an event. Hence, this constraint is not specific to a given time, and since we cannot determine which resource is assigned to an event resource, we cannot evaluate this constraint. Therefore our approach is heuristic in nature for XHSTT instances which use this constraint for free resources. However, since a penalty of zero is imposed in Stage I, the lower bound on Stage I is also a lower bound on the original model.
- *Avoid Clashes*: This constraint penalizes resources which are assigned to more than one event at each time, and applies to the entire set of events $\mathcal{E}$, and to a set of resources $R_c^{\text{avoidclashes}} \subseteq \mathcal{R}$. Unfortunately, we see no way of modeling this constraint accurately in the graph G. However, the penalty is approximated by adding a dummy-vertex for every resource $r \in R_c^{\text{avoidclashes}}$, which have the same adjacent edges as the vertex of the resource itself. The weight of the edge is the weight of the *Avoid Clashes* constraint $c \in \mathcal{C}$. Section 9.4.2 will show that this approximations in in fact an exact approach for the considered instances.
- *Avoid Unavailable Times*: This constraint $c \in \mathcal{C}$ specifies that certain resources $R_c^{\text{avoidunavailable}}$ are unavailable at certain times $T_c^{\text{avoidunavailable}}$, and penalizes the assignment of resources to these times. This is handled by issuing a separate version of G for every time, denoted G_t. For every *Avoid Unavailable Times* constraint $c \in \mathcal{C}$ and for every time $t \in T_c^{\text{avoidunavailable}}$ it applies to, add the penalty w_c to all edges adjacent to the resources $R_c^{\text{avoidunavailable}}$ in the graph G_t.

Since the bipartite graph now contains edge-weights, we seek a *minimum weight maximum matching*, i.e. a maximum matching which assigns (sub-event, event resource) elements to resources such that the summed weight on the used edges is minimal. However, since the *Avoid Split Assignments* constraints can not be handled, the value of the matching is only a lower bound on the actual penalty (for instances which actually use this constraint). The quality of this lower bound depends on the dataset in question. Theorem 4.2 of Sørensen and Dahms (2014) states the following.

Theorem 9 (Lower bound on minimum weight maximum matching)**.** *Given a bipartite graph $G = (A \cup \mathcal{R}, E)$ with edge weights, where $\mathcal{W}$ denotes the set of all edge weights in G sorted from smallest to largest, $G_{\leq w}$ is the subgraph of G containing only edges with weight $\leq w \in \mathcal{W}$, and $a_w \in \mathbb{N}_0$ defined as*

$$a_w = \begin{cases} |A| - \mathit{def}(G_{\leq w}) & \mathit{ord}(w) = 0, \\ \mathit{def}(G_{\leq w'}) - \mathit{def}(G_{\leq w}) & \mathit{ord}(w) > 0, \mathit{ord}(w') = \mathit{ord}(w) - 1, \end{cases} \tag{9.3}$$

then $\sum_{w \in \mathcal{W}} a_w$ is a lower bound on the minimum weight maximum matching in G.

Thereby we can approximate the penalty of the resource constraints in Stage I of the decomposition. This requires the following variables; $\text{def}_{t,\leq w} \in \mathbb{R}^+$ is the deficiency of subgraph

$G_{t,\leq w}$ of graph G_t for time $t \in \mathcal{T}$, and $a_{t,w} \in \mathbb{R}^+$ is the variable associated with Theorem 9 for time $t \in \mathcal{T}$ and weight $w \in \mathcal{W}$. The parameter $\Gamma_{t,\leq w}(S)$ defines the neighbors for set $S \subseteq A$ of subgraph $G_{t,w}$.

Models (9.4) and (9.5) shows Stage I and Stage II, respectively. In the original model eqs. (9.1e) to (9.1h) define that the duration of the active sub-events must match the duration of the equivalent event. Since we cannot determine which events will be assigned free resources, these constraints are simplified in Stage I. The imposed condition is that a sub-event is active if it is assigned a time, contrary to also involving the assignment of free resources. Constraint (9.4c) defines this requirement. Whether this is a limitation of the model is discussed in Section 9.4.2. As described above, only a subset of native XHSTT constraints $\mathcal{C}$ can be optimally handled in Stage I. These are denoted $\mathcal{C}^I$, and are defined by Constraint (9.4g). Equations (9.4d) to (9.4f) along with the additional term in the objective function define the lower bound on the constraints $\mathcal{C}^h$. Notice that all variables except $y_{se,t}$ can be stated as continuous, as they will naturally take integer values.

Given the solution for Stage I $y^*_{se,t}$, a solution to the full model can be derived by Stage II. Constraint (9.5b) ensures that each event resource is assigned one resource (possibly the dummy resource).

Stage I (9.4)

$$\min \quad z^I = \sum_{c \in \mathcal{C}^I} w_c \cdot \text{CostFunction}(s_{c,p}) + \sum_{t \in \mathcal{T}, w \in \mathcal{W}} a_{t,w} \tag{9.4a}$$

s.t.

$$\sum_{t \in \mathcal{T}} y_{se,t} = 1 \qquad \forall se \in \mathcal{SE} \tag{9.4b}$$

$$\sum_{se \in e, t \in \mathcal{T} \setminus t_D} y_{se,t} = D_e \qquad \forall e \in \mathcal{E} \tag{9.4c}$$

$$\sum_{se \in S, t' \in T^{\text{start}}_{se,t}} y_{se,t'} - \text{def}_{t,\leq w} \leq |\Gamma_{t,\leq w}(S)| \quad \forall S \subseteq A_t, t \in \mathcal{T}, w \in \mathcal{W} \tag{9.4d}$$

$$\sum_{se \in A_t, t' \in T^{\text{start}}_{se,t}} y_{se,t'} - \text{def}_{t,\leq w} = a_{t,w} \qquad \forall t \in \mathcal{T}, w \in \mathcal{W}, \text{ord}(w) = 0 \tag{9.4e}$$

$$\text{def}_{t,\leq w'} - \text{def}_{t,\leq w} = a_{t,w} \qquad \begin{array}{l} \forall t \in \mathcal{T}, w, w', \in \mathcal{W}, \text{ord}(w) > 0, \\ \text{ord}(w') = \text{ord}(w) - 1 \end{array} \tag{9.4f}$$

$$(y_{se,t}, s_{c,p}) \in \mathcal{C}^I \tag{9.4g}$$

$$y_{se,t} \in \{0, 1\} \tag{9.4h}$$

$$\text{def}_{t,\leq w}, a_{t,w}, s_{c,p} \in \mathbb{R}^+ \tag{9.4i}$$

Stage II (9.5)

$$\min \quad z^{II} = \sum_{c \in \mathcal{C}} w_c \cdot \text{CostFunction}(s_{c,p}) \tag{9.5a}$$

s.t.

$$\sum_{r \in \mathcal{R}} w_{se,er,r} = 1 \; \forall se \in \mathcal{SE}, er \in se \tag{9.5b}$$

$$(w_{se,er,r}, y^*_{se,t}, s_{c,p}) \in \mathcal{C} \tag{9.5c}$$

$$w_{se,er,r} \in \{0,1\} \tag{9.5d}$$

$$s_{c,p} \in \mathbb{R}^+ \tag{9.5e}$$

9.4.2 Practical Considerations

The described TSD has some limitations, described below in details. The impact of each limitation in practice is discussed.

- Recall that a sub-event is active if it is assigned a time and/or a non-preassigned resource. Constraint (9.1h) of the original MIP model ensures that the sum of the duration of all active sub-events of an event must equal the duration of the event. This raises an issue in the decomposition, since it might be optimal to leave a sub-event inactive by not assigning it to a time in Stage I, and only making it active by assigning a resource to one of its event-resources. However, we consider this scenario unlikely, on the basis of the following: By investigation of the available XHSTT instances, every event not preassigned to a time is considered by an *Assign Time* constraint, and this constraint is a hard constraint. This means that all such events request an assignment of a time. Likewise, all free event resources are considered by a hard *Assign Resource* constraint. This means that if for some instance a solution with 0 violations of hard constraints is known, then it cannot be optimal to not assign times to sub-events for this event. Furthermore, suppose that some event resource is free, its *Assign Resource* constraint has weight w_1, and its *Assign Time* constraint that applies to the event has weight w_2. If a solution is known with violation of hard constraints less than $w_2 - w_1$ then the TSD will be an exact approach in this aspect. By investigation of the XHSTT instances, it is shown that this limitation only has impact the instances DenmarkFalkonergaardensGymnasium2012, DenmarkHasserisGymnasium2012 and DenmarkVejenGymnasium2009. Thereby lower bounds cannot be generated for these instances, so optimality cannot be guaranteed theoretically.
- The lower bound on constraints $\mathcal{C}^h$ assumes that the CostFunction for these constraints is *Sum*, as the nonlinear CostFunctions requires more detailed information on the specific origin of the constraint violations. For instance, a nonlinear CostFunction for an *AssignResource* constraint requires knowledge on which specific sub-events and event resources that are not assigned a resource, contrary to only knowledge of the magnitude of the violation as determined by the lower bound. However in practice, all XHSTT instances encountered use the *Sum* CostFunction for all constraints $\mathcal{C}^h$. Thereby this is not an issue in practice.
- Constraints *Limit Idle Times*, *Cluster Busy Times*, *Limit Busy Times*, *Limit Workload* and *Avoid Split Assignment* cannot be handled optimally for free resources by the decomposition. Table 9.1 shows the encountered XHSTT instances with at least one free resource, and the constraints of each instance which applies to at least one free resource. The table shows that no instance contains *Limit Idle Times* or *Cluster Busy Times* constraints which

apply to free resources, so these constraints can be optimally handled by the decomposition for the considered instances. In case of *Limit Busy Times*, *Limit Workload* and *Avoid Split Assignment*, the same three instances contain such constraints which applies to free resources. Therefore the decomposition cannot guarantee optimally for these instances, but is though capable of providing lower bounds. For the remaining instances, these constraints can be handled optimally.

- The *Avoid Clashes* constraint can only be approximated in theory. However, practical considerations of the XHSTT instances have shown the following two observations: 1) No two *Avoid Clashes* constraints apply to the same resource, and 2) All *Avoid Clashes* and *Assign Resource* constraints are hard constraints, and have a weight of value 1, except in the three Australian instances. These observations mean that in terms of these constraints, equivalent cost arises given the same amount of violation of either of them (since we also assume CostFunction *Sum* for these constraints, as previously described). Therefore these constraints be modeled accurately in the bipartite graph by introducing a dummy-vertex for each event-vertex, with a single edge of weight 1 which is adjacent to the event-vertex. However, the following approach is more efficient in practice. Denote by $a_t^{\text{clash}} \in \mathbb{N}_0$ the amount of violation of all *Avoid Clashes* and *Assign Resource* constraints in time $t \in \mathcal{T}$. Constrain this variable by

$$\sum_{se \in A_t, t' \in T_{se,t}^{\text{start}}} y_{se,t'} - \sum_{w \in \mathcal{W}} a_{t,w} = a_t^{\text{clash}} \qquad \forall t \in \mathcal{T}, \tag{9.6}$$

  and penalize it in the objective function accordingly. By this approach, the size of the bipartite graph is significantly reduced.

To summarize these considerations, all XHSTT instances, except the three Australian and the three Danish, can be solved optimally in theory by the decomposition, if we were capable of determining the optimal matching of sub-events to resources in Stage I. However, since we only know a lower bound on the matching, optimality cannot be guaranteed. In favor of the decomposition is the fact that the total amount of variables in the problem is considerably reduced, which might be beneficial for the MIP solver. In case of the Australian instances, lower bounds can be generated. In case of the Danish instances, their structure prevent the generation of lower bounds.

9.4.3 Generating Hall Inequalities

Constraint (9.4d) contains an exponential amount of inequalities, so in practice it is intractable to generate them all. However, not all of these inequalities are necessary for maintaining optimality. In fact, in practice only a small subset is required. Our approach for generating these necessary inequalities is domain-specific, and based on characteristics of the bipartite graphs of the XHSTT instances. We already argued that most XHSTT instances contain only one *Assign Resource* constraint and one *Avoid Clashes* constraint. What remain to be considered is therefore *Prefer Resources* and *Avoid Unavailable Times*. In the following, an informal explanation of our approach to generate these inequalities is given. For formal proofs of the following claims for the general case of bipartite graphs, see Sørensen and Dahms (2014).

Consider a subset $a \subseteq A_t$ for some time $t \in \mathcal{T}$. If all sub-events of a are adjacent to the exact same set of rooms with the same edge-weights, then it is easy to see that only one inequality is required, namely the one for a itself. Furthermore, consider two subsets $a_1, a_2 \subseteq A_t$ with the same property. In this case, three inequalities are required, namely those of a_1, a_2 and $a_1 \cup a_2$. Using

this analogy of grouping of elements of A_t all required inequalities can be generated. However, the amount of inequalities is exponential in the number of groups, and the number of groups might not be small. In our terminology, a *Prefer Resources* constraint alters the weight on edges between a certain set of events and a certain set of resources, and an *Avoid Unavailable Times* constraint alters the weight on all edges adjacent to a certain set of resources (since we consider the graph for a specific time). Hence if a XHSTT instance contains few of these constraints, it is reasonable to expect that the amount of groups of (event,event-resource) elements is low.

Another common property of the bipartite graph which we exploit is the following: Many event-resources are fixed to a certain resource, and therefore only contain one edge in the graph. Such event-resources can be excluded from the grouping of elements of A_t, and handled implicitly using auxiliary variables, see (Sørensen and Dahms, 2014, Theorem 4.5).

By these considerations, it has been possible to generate the necessary Hall inequalities for most XHSTT instances. We will assume that the presented approach is intractable in case more than 10 groups of resources are needed. In these cases the groups are ranked based on amount of adjacent edges to the other groups, and the inequalities are generated for the top 10 groups. This limitation was necessary for four instances, see Table 9.2.

Table 9.1: Constraints of XHSTT instances in the archive `ALL_INSTANCES` which applies to events with free resources or resource which are free for at least one event. "Y" means that the instance contains at least one of such constraints which applies to at least one event with at least one free resource.

	AustraliaBGHS98	AustraliaSAHS96	AustraliaTES99	DenmarkFalkonergaardensGymnasium2012	DenmarkHasserisGymnasium2012	DenmarkVejenGymnasium2009	EnglandStPaul	FinlandArtificialSchool	NetherlandsKottenpark2003	NetherlandsKottenpark2005	NetherlandsKottenpark2009	SpainSchool
Assign Resource	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Assign Times	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Avoid Clashes	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Avoid Split Assignment	Y	Y	Y	-	-	-	-	-	-	-	-	-
Avoid Unavailable Times	Y	Y	Y	-	-	-	-	-	Y	Y	Y	Y
Cluster Busy Times	-	-	-	-	-	-	-	-	-	-	-	-
Distribute Split Events	Y	Y	-	-	-	-	-	-	-	-	-	-
Link Events	Y	Y	Y	Y	Y	Y	Y	-	Y	Y	Y	-
Limit Busy Times	Y	Y	Y	-	-	-	-	-	-	-	-	-
Limit Idle Times	-	-	-	-	-	-	-	-	-	-	-	-
Limit Workload	Y	Y	Y	-	-	-	-	-	-	-	-	-
Order Events	-	-	-	-	-	-	-	-	-	-	-	-
Prefer Resources	Y	Y	Y	Y	Y	Y	-	Y	Y	Y	Y	Y
Prefer Times	Y	Y	-	-	-	-	-	Y	Y	Y	Y	-
Split Events	Y	Y	-	-	-	-	-	Y	Y	Y	Y	Y
Spread Events	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

9.5 Solution Method

Given the structure of the XHSTT objective, solutions are always ranked first on the hard cost, and secondly on the soft cost. Kristiansen et al. (2013) propose a two-step solution method which first solves a partial model containing only the constraints and variables necessary to determine the hard cost. This is denoted Step I. Once Step I is solved, the hard cost of the final solution is known. Now the model is extended with variables and constraints necessary to determine the soft cost. This is denoted Step II. The same approach is used in each stage of the TSD, as described in the following. First the Stage I model is constructed such that it only contains hard constraints. This model represents Step I of Stage I and is denoted $M^{\text{StageI}}_{\text{StepI}}$. $M^{\text{StageI}}_{\text{StepI}}$ is solved given some timelimit, to obtain a solution which represents the hard cost. The obtained lower bound is a lower bound on the hard cost for the original problem.

$M^{\text{StageI}}_{\text{StepI}}$ is now extended with soft constraints as well, to obtain model $M^{\text{StageI}}_{\text{StepII}}$. In case model $M^{\text{StageI}}_{\text{StepI}}$ was solved to optimality, the violation of the hard constraints of model $M^{\text{StageI}}_{\text{StepII}}$ must equal the violation of the hard constraints of model $M^{\text{StageI}}_{\text{StepI}}$, which is ensured by an equality constraint on the relevant variables. The lower bound obtained is an lower bound on the soft cost of the overall problem. In case model $M^{\text{StageI}}_{\text{StepI}}$ was not solved to optimality, all variables in model $M^{\text{StageI}}_{\text{StepII}}$ are fixed to their equivalent value in model $M^{\text{StageI}}_{\text{StepI}}$ to obtain the soft cost of the solution. The reasoning is that the algorithmic time should be spend on minimizing the hard cost defined by model $M^{\text{StageI}}_{\text{StepI}}$. So if this model is not solved to optimality, the MIP solver is only invoked on model $M^{\text{StageI}}_{\text{StepII}}$ to obtain the soft cost.

In case the dataset in question has no free resources, the solution to Stage I is also a solution to the original problem, meaning that Stage II can be skipped. In case the dataset contains free resources, the solution of Stage I is given as input to Stage II, and the solution procedure continues. Model $M^{\text{StageII}}_{\text{StepI}}$ is build, which contains all hard constraints of the dataset. The two-step procedure of Stage II is analogous to that of Stage I, except that Stage II does not generate lower bounds on the cost. The output of Stage II is a solution to the original model.

Figure 9.2 illustrates the solution method.

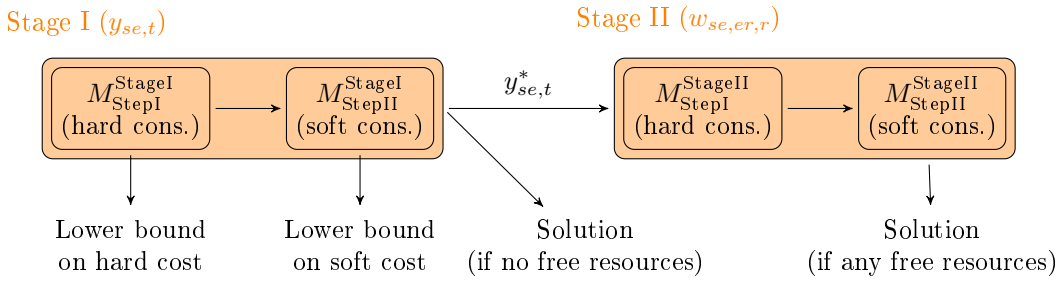


Figure 9.2: Solution method outline.

9.6 Computational Results

To evaluate the TSD, empirical results are established on a number of XHSTT instances. Specifically, the XHSTT archive `ALL_INSTANCES` is used, which contains 38 non-artificial instances. During the computational experiments the following observations has been made for instances with only fixed resources (recall that in such cases, only Stage I is needed in the solution process):

- The size of the original MIP model (9.1) and the Stage I model (9.4) are roughly the same after presolve.
- By inspection of the presolved model of Stage I, all $x_{se,t,er,r}$ variables have been removed from the model.
- The performance of the TSD is roughly equal to that of the original MIP.

In the following an argument for this behavior is attempted. Given is a dataset with all resources are fixed. This means that for sub-event $se \in \mathcal{SE}$ and time $t \in \mathcal{T}$, indices er and r can be dropped from $x_{se,t,er,r}$. Furthermore, all variables $w_{se,er,r}$ are fixed to value 1 by definition. Therefore the Stage I model (9.4) will closely resemble the original MIP model (9.1) in these particular cases. However, due to the heuristic behavior of presolve of the MIP solver, it cannot be guaranteed that these exact reduction steps is performed. But as the observed sizes of the models are roughly equal, and the models perform similarly, only performance for instances with free resources are reported in the following.

Table 9.2: Comparison of solution methods for instances in archive `ALL_INSTANCES` with free resources. Columns $Time_1$ and $Time_2$ indicate the time spend on Step I and Step II, respectively. For both solution approaches is shown the obtained lower bound *LB* and the objective value of the obtained solution *Obj*. In case a lower bound or a solution was not found, a dash "-" is written. Columns *Best* and *Best LB* indicate the best known solution and lower bound, respectively.

		Original MIP model				TSD							
						Stage I		Stage II					
	Instance	$Time_1$	$Time_2$	LB	Obj	$Time_1$	$Time_2$	$Time_1$	$Time_2$	LB	Obj	Best	Best LB
AU	BGGHS98	>21600	-	(-, -)	(-, -)	20520	-	-	-	(0, -)	(-, -)	(3, 494)	(0, 0)
AU	SAHS98	>21600	-	(-, -)	(-, -)	20520	-	-	-	(0, -)	[2](-, -)	(8, 52)	(0, 0)
AU	TES99	>21600	-	(-, -)	(-, -)	20520	-	1	1	(0, -)	[2](162044, 85)	(1, 140)	(0, 0)
DK	Falkoner2012	>21600	-	(0, -)	(341106, 9951)	20520	-	4	3	(0, -)	[2](3981, 14196)	(2, 23705)	(0, 0)
DK	Hasseris2012	>21600	-	(7, -)	(869471, 14414)	20520	-	6	7	[1](102, -)	(5850, 38941)	(293, 32111)	(0, 0)
DK	Vejen2009	>21600	-	(2, -)	(928, 8078)	20520	-	2	2	[1](2, -)	[2](3344, 20643)	(20, 18966)	(2, 0)
ENG	StPaul	>21600	-	(0, -)	(1227, 0)	20520	-	2	2	(0, -)	(80783, 10516)	(0, 136)	(0, 0)
FI	ArtificialSchool	>21600	-	(0, -)	(12, 31)	669	19851	0	0	(0, 0)	(0, 1)	(0, 0)	(0, 0)
NL	Kottenpart2003	>21600	-	(-, -)	(-, -)	20520	-	2	2	(0, -)	(11398, 3660)	(0, 1410)	(0, 0)
NL	Kottenpart2005	>21600	-	(-, -)	(1617, 68)	20520	-	3	3	(0, -)	(13341, 672)	(0, 1078)	(0, 9)
NL	Kottenpart2009	>21600	-	(0, -)	(12461, 20)	20520	-	3	2	(0, -)	(118236, 1845350)	(0, 9035)	(0, 160)
ES	School	6125	>15475	(0, 309)	(0, 6818)	2246	18274	0	0	(0, 331)	(0, 718)	(0, 357)	(0, 330)

[1] It cannot be guaranteed that this is a true lower bound. See Section 9.4.2.
[2] An artificial limit on the amount of Hall inequalities was imposed at least one time.

		Step I of org. MIP model			TSD $M_{\text{StepI}}^{\text{StageI}}$		
	Instance	Vars	Cons	Non-zeros	Vars	Cons	Non-zeros
AU	BGGHS98	$1.2 \cdot 10^7$	$5.5 \cdot 10^5$	$6.8 \cdot 10^7$	$1.6 \cdot 10^5$	$1.8 \cdot 10^5$	$1.3 \cdot 10^7$
AU	SAHS98	–	–	–	–	–	–
AU	TES99	$8.0 \cdot 10^5$	$1.1 \cdot 10^5$	$4.1 \cdot 10^6$	$6.2 \cdot 10^4$	$9.1 \cdot 10^4$	$1.2 \cdot 10^7$
DK	Falkoner2012	$8.5 \cdot 10^4$	$1.4 \cdot 10^4$	$3.3 \cdot 10^5$	$7.6 \cdot 10^3$	$4.2 \cdot 10^3$	$1.2 \cdot 10^5$
DK	Hasseris2012	$4.6 \cdot 10^6$	$2.6 \cdot 10^5$	$1.8 \cdot 10^7$	$1.5 \cdot 10^5$	$1.5 \cdot 10^5$	$1.6 \cdot 10^7$
DK	Vejen2009	$6.6 \cdot 10^6$	$4.1 \cdot 10^5$	$2.5 \cdot 10^7$	$2.3 \cdot 10^5$	$2.3 \cdot 10^5$	$4.4 \cdot 10^6$
ENG	StPaul	$3.7 \cdot 10^6$	$1.9 \cdot 10^5$	$1.4 \cdot 10^7$	$1.0 \cdot 10^5$	$6.4 \cdot 10^4$	$1.6 \cdot 10^6$
FI	ArtificialSchool	$2.4 \cdot 10^6$	$1.6 \cdot 10^5$	$9.2 \cdot 10^6$	$4.7 \cdot 10^4$	$3.5 \cdot 10^4$	$3.8 \cdot 10^5$
NL	Kottenpart2003	$1.9 \cdot 10^6$	$2.0 \cdot 10^5$	$7.5 \cdot 10^6$	$1.4 \cdot 10^5$	$1.3 \cdot 10^5$	$2.8 \cdot 10^6$
NL	Kottenpart2005	$2.9 \cdot 10^6$	$2.7 \cdot 10^5$	$1.1 \cdot 10^7$	$1.8 \cdot 10^5$	$1.8 \cdot 10^5$	$3.6 \cdot 10^6$
NL	Kottenpart2009	$3.3 \cdot 10^6$	$2.0 \cdot 10^5$	$1.3 \cdot 10^7$	$9.1 \cdot 10^4$	$6.9 \cdot 10^4$	$3.9 \cdot 10^6$
ES	School	$5.6 \cdot 10^4$	$1.9 \cdot 10^4$	$3.2 \cdot 10^5$	$3.8 \cdot 10^4$	$2.0 \cdot 10^4$	$8.1 \cdot 10^5$

Table 9.3: Comparison of models Step I of original MIP model and $M_{\text{StepI}}^{\text{StageI}}$ model of TSD after presolve performed by Gurobi. Columns *Vars*, *Cons*, *Non-zeros* denotes the number of variables, constraints and non-zeros, respectively. Instance AU SAHS98 cannot be presolved within a 48 hour time-limit, so results for this instance are not reported.

All experiments were performed on machines with an Intel Core i7 CPU clocked at 2.80GHz and 12 GB of RAM, running Windows 8 64bit. Gurobi 5.5.0 was used as MIP solver, which is one of the best performing general-purpose MIP solvers available (see Mittelman (2013)). On this particular CPU, Gurobi uses eight threads. By default, Gurobi uses the dual simplex algorithm to solve the root LP-relaxation. A study of the performance of Gurobi for the XHSTT models has indicated that it is usually advantage to instead use the `concurrent` method, which uses one thread for the primal simplex algorithm and one thread for the dual simplex algorithm, and the remaining threads (six in this case) are used by the barrier algorithm. We use this parameter setting for all experiments.

To evaluate the performance of the TSD, the following test setup is used. A time-limit of 6 hours (21600 seconds) is imposed for both the original MIP model and the TSD. In case of the TSD, the time-limit for Stage I is set to $0.95 \cdot 21600s = 20520s$, as it is expected that Stage I is way harder to solve than Stage II. Thereby the time-limit for Stage II is 1080s. In case Stage I is solved to optimality, the excess time is added to the time-limit of Stage II.

Table 9.2 shows the obtained results. The TSD produces the best solution in 6 cases, and the MIP model is best in 6 cases. It was expected that the advantage of the TSD would be more pronounced. Table 9.2 shows that the majority of time is spend in Step I for the MIP model, and in model $M_{\text{StepI}}^{\text{StageI}}$ for the TSD. To further compare the differences between the solution approaches, Table 9.3 shows statistics of these models after presolve has been invoked by the MIP solver. The table shows that the $M_{\text{StepI}}^{\text{StageI}}$ model is significantly smaller than the Step I model of the original MIP model. Therefore it is surprising that the TSD does not significantly outperform the original MIP model. When investigating the logs of the MIP solver for the TSD, it is seen that the solver seems to spend a lot of time improving the LP-relaxation in the rootnode. In many cases the total amount of nodes explored throughout the solution process is very small (for the larger instances, work is only performed on the rootnode). This suggests that the instances are a significant challenge for the MIP solver, even for the smaller models produced by the TSD. It is remarked that for two of the smallest instances (FI ArtificialSchool and ES School), the $M_{\text{StepI}}^{\text{StageI}}$ model is solved to optimality, and in these cases the TSD outperforms the

original MIP model in terms of both solution- and lower bound quality. This suggests that the potential of techniques based on TSD is large, but the approach presented in this paper is not sufficient for obtaining good results in all cases.

In all cases are the root LP-relaxation of the $M_{\text{StepI}}^{\text{StageI}}$ model solved in the TSD (indicated by the *LB* column). For the MIP model. this is only the case for 7 instances. All lower bounds produced by the TSD are better or equivalent to those produced by the MIP model. Table 9.2 also shows that the obtained solutions are far from the best known ones. This indicates that significant work is still needed on exact methods before these can compete with heuristics for this particular optimization problem.

9.7 Conclusion

A Two-Stage Decomposition (TSD) for the generalized High School Timetabling Problem has been proposed, based on a Mixed-Integer Programming (MIP) formulation of the general XHSTT format. The approach is innovate and it has been shown that in theory the decomposition is advantageous to solve compared to the original MIP model.

The XHSTT instances can be thought of as two distinct groups, those where all resources are preassigned (fixed resources), and those where at least one event expects the assignments of some resource of a certain type (free resources). For an instance with only preassigned resources, the decomposition model roughly resembles that of the original MIP model, and no significant difference was observed in computational experiments. For an instance with free resources, the decomposition yields significantly smaller models, which is an advantage in theory. However, for the tested instances (a total of 12 of the considered instances have free resources), the decomposition did not produce as good solutions and lower bounds as was expected on beforehand.

Even though TSD is a promising method in theory, many of the tested instances are too large to be handled effectively. We believe that the decomposition is advantageous for these instances, but still inadequate for establishing state-of-the-art results (at least with the used test-setup). This is indicated by the fact that for two of the smallest instances with free resources, the computational results show that the decomposition outperforms the original MIP model. To enhance the decomposition, further steps are required. This could for instance be other types of decomposition which could be applied to the MIP models yielded by the decomposition, such as Dantzig-Wolfe Decomposition.

Another topic for future research is more theoretical insight in the decomposition, both w.r.t. deriving a better bound (or even optimal solution) of the matching in the bipartite graph modeling the anonymous assignment of resources, and w.r.t. an algorithm for generating the necessary Hall inequalities. Such improvements would theoretically strengthen the decomposition. Furthermore, the application of TSD to the XHSTT format could be improved in a number of ways (see Section 9.4.2). Ultimately, this would lead to the possibility of generating lower bounds for any XHSTT instances independently of the characteristics of the particular instance.

Bibliography

M. ter Braak. A hyperheuristic for generating timetables in the xhstt format. Master's thesis, University of Twente, June 2012.

E. Burke, J. Marecek, A. Parkes, and H. Rudová. Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 37(3):582–597, 2010.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

S. Daskalaki and T. Birbas. Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160(1):106 – 120, 2005. ISSN 0377-2217.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

J.-K. Hao and U. Benlic. Lower bounds for the itc-2007 curriculum-based course timetabling problem. *European Journal of Operational Research*, 212(3):464 – 472, 2011. ISSN 0377-2217.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Integer programming for the generalized (high) school timetabling problem. *Journal of Scheduling*, Submitted 5/9-2013, 2013.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

H. Mittelman. Benchmarks for optimization software. `http://plato.asu.edu/bench.html` [Accessed 28/10-2013], 2013.

M. Pimmer and G. R. Raidl. A timeslot-filling heuristic approach to construct high-school timetables. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Advances in Metaheuristics*, volume 53 of *Operations Research/Computer Science Interfaces Series*, pages 143–157. Springer New York, 2013. ISBN 978-1-4614-6321-4.

G. Post. Benchmarking project for (high) school timetabling. `http://www.utwente.nl/ctit/hstt/` [Accessed 28/10-2013], 2013.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012. ISSN 0254-5330.

J. Romrös and J. Homberger. An evolutionary algorithm for high school timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 485–488. SINTEF, 2012.

H. Santos, T. Toffolo, S. Ribas, and R. Gomes. Integer programming techniques for the nurse rostering problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

M. K. Y. Shambour, A. T. Khader, A. Kheiri, and E. Özcan. A two stage approach for high school timetabling. In M. Lee, A. Hirose, Z.-G. Hou, and R. Kil, editors, *Neural Information Processing*, volume 8226 of *Lecture Notes in Computer Science*, pages 66–73. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-42053-5.

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

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

Chapter 10 Paper E

A Matheuristic for High School Timetabling

Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract An emerging branch within Operations Research is the hybridization of *Metaheuristics* and *Mathematical Programming*, commonly known as *matheuristics*. This paper presents a matheuristic for two versions of the high school timetabling problem, the generalized problem based on the XHSTT format, and the Danish case, respectively. The algorithm is build on top of a *Mixed-Integer Linear Programming* (MILP) formulation (distinct formulations are used for the two versions of the problem), such that the feasible area is explored iteratively using a state-of-the-art MILP solver. Specifically, the algorithm iteratively considers a (small) part of the MILP model (denoted a *subproblem*), and attempts to improve the current best solution by altering the values of the respective variables of the considered part. The matheuristic is in principle applicable for general MIPs, but we have chosen to build the subproblems using problem-specific knowledge to improve performance. Extensive computational results are established, and good performance is shown for both versions of the problem; for the generalized problem, the matheuristic is competitive with the winner of round 2 of the International Timetabling Competition 2011. For the Danish version of the problem, the matheuristic performs best compared to three other algorithms, given a large timelimit. Given a smaller timelimit, the matheuristic is outperformed by another heuristic.

10.1 Introduction

For many practical optimization problems a full-scale exact solution approach is computational intractable, and (meta)heuristics have proven capable of providing good solutions. An emerging branch within Operations Research is the hybridization of these two classes of methods, commonly known as *matheuristics*. Both of these classes of methods has specific advantages; Heuristics are usually less sensitive to the size of the input data, whereas mathematical programming techniques are theoretically capable of finding optimal solutions. By hybridization, it is the desire to *enjoy the best from both worlds* (Ryan (2012)).

In this paper two versions of the *High School Timetabling Problem* (HSTP) are considered. A *Mixed-Integer Linear Programming* (MIP) formulation is known for both these test-cases, which lies as a foundation for our approach. A matheuristic for this problem is constructed by iteratively solving an *artificially-constrained* version of the problem (from now on known as a *subproblem*) by the MIP solver. Specifically, the solution space is reduced considerably by allowing the MIP solver to change the value of only a subset of variables in each iteration. This

results in MIPs which are considerably simpler than the original MIP, and which can be solved effectively by the MIP solver. In the following, the size of a subproblem refers to the amount of free variables.

Selecting the artificial constraints to add to the model poses a trade-off; On one hand, the constraints should make the problem solvable by the solver. On the other hand, the constraints should allow as much freedom as possible, such that improving solutions can be found. Clearly, what makes a good subproblem w.r.t. these aspects might differ among problem instances. Therefore we propose to dynamically adjust the size of sub-problems based on previous performances of the solver for the particular problem-instance. Hence, if a given problem-instance is 'easy', then the size of subproblems will be increased by the adaptive layer of the heuristic, giving more degrees of freedom to the solver. If applicable, subproblems might be expanded to full problem-size, and the approach will be exact. If a given problem-instance is 'hard', subproblems will be down-sized accordingly.

Our approach also considers various ways of constructing subproblems. This means that a way of selecting a subproblem to use is needed, which is another feature of the adaptive layer. Specifically, we track the total improvement made by the subproblem to the objective so far, as well as the total time spend. On the basis of these indicators, a mechanism for subproblem-selection is imposed which selects the most favorable subproblems often.

Even though a process like the one described could be designed to give optimal solutions, that is not the focus of this paper. Instead we aim at an algorithm which finds good incumbent solutions, by heuristically constructing subproblems and not requiring to solve these to optimality. Furthermore, the general matheuristic described in this paper is not problem-specific. Problem-specific knowledge is used to construct the subproblems, but this should be easily adaptable to other (timetabling) optimization problems.

The paper is organized as follows; Section 10.2 describes related approaches from the literature. Section 10.3 describes the matheuristic in details. Section 10.4 describes both versions of the high school timetabling problem in details. Section 10.5 shows the obtained computational results. Section 10.6 concludes on our findings.

10.2 Related work

In terms of high school timetabling in general, we refer to Pillay (2013) for a recent survey on different solution methodologies. In the following we briefly survey methods related to the described matheuristic.

The *Corridor Method* (Sniedovich and Voß, 2006) is similar to our approach in that exogenous constraints are imposed on the optimization problem at hand, to make it tractable by a given exact optimization method. The method resides on *method-based neighborhoods*, implying that the subproblems should be constructed such that they are easily solvable by the exact optimization method. This is in contrary to *move-based neighborhoods*, in which very small changes is made to a given feasible solution.

Large optimization problems can often be seen as being composed of different parts. The idea of POPMUSIC (or *Partial Optimization Meta-heuristic under Special Intensification Conditions*) is to locally optimize sub-parts of a solution in an iterative fashion (Voss (2001),Taillard and Voss (2002)). The method is stopped once optimal solutions are found for all sub-parts, contrary to the concept of our approach.

Our approach can be seen as a heuristic version of *Local Branching* (Fischetti and Lodi, 2003). Local Branching is a general technique for solving MIPs with binary variables, in which linear inequalities are used to control the number of changes k made to the current solution. Branching

on $\leq k$ and $\geq k+1$ respectively, it is then the desire to quickly achieve good incumbent solutions. Instead of a general methodology for MIPs, our method uses problem dependent knowledge in selecting the variables of subproblems.

A notable contribution within the area of matheuristics is Maniezzo et al. (2009a), which covers material presented at the *Matheuristics 2008* workshop. Blum et al. (2011) survey different branches within hybrid metaheuristics, dividing the algorithms into five different categories. Prandtstetter and Raidl (2008) hybridize Integer Linear Programming and Variable Neighborhood Search for a car sequencing problems, such that large neighborhoods are searched by a MIP solver when smaller heuristic-based neighborhoods was unable to improve a solution further. Another noteworthy contribution is Maniezzo et al. (2009b) which contains contributions from the *Matheuristics workshops*.

Solving timetabling problems by exact methods is not very common in the literature. One exception is Avella et al. (2007) which presents a MIP model for a case of high school timetabling, and solves this MIP within a *Very-Large Neighborhood Search* (VLSN) algorithm. Such a type of algorithm indeed fits within the matheuristic paradigm. In the VLSN algorithm of Avella et al. (2007) a timetable is iteratively improved by fixing the schedule for all teachers except two. This is quite similar to our approach, except we do not restrict the algorithm to choose any specific number of teachers, nor to any specific type of resource. In fact, the number of resources which can have their timetable altered in an iteration is adaptively adjusted for better performance. Unfortunately the datasets which Avella et al. (2007) use to establish computational results are not available in a standard format, so a comparison of empirical results cannot be made.

Our approach can be seen as being related to *Large Neighborhood Search* (LNS) (Pisinger and Ropke, 2010), and the variant *Adaptive Large Neighborhood Search* (ALNS). In a LNS algorithm, a solution is iteratively changed using *destroy* and *repair* operators. In a ALNS algorithm, multiple destroy/repair operators are used, and those that perform well are used most often. Sørensen and Stidsen (2013b) apply ALNS to the Danish case of high school timetabling.

10.3 Matheuristic

The key idea of our approach is to iteratively improve a solution by exploring neighbor-solutions, as is done in *local search* methodologies. We say that we explore the *neighborhood* of a solution, and this is done by a MIP solver. Multiple neighborhoods are used, denoted by the set N, and we impose an adaptive mechanism to select the neighborhood structures which have previously performed best. Furthermore it is assumed that a solution S can be fully described by the values of the decision variables $\mathcal{X}$. This should be understood such that a setting of values in $\mathcal{X}$ implicitly define the values for all other variables by means of the constraints of the MIP, and thereby $\mathcal{X}$ are sufficient for describing a solution to the problem instance at hand. Note that we pose no restrictions on how the variables $\mathcal{X}$ are defined; They can be defined as integer or continuous as desired.

A neighborhood $n \in N$ is essentially a method that selects a subset of variables $X \subseteq \mathcal{X}$. This neighborhood is explored by fixing all other variables $\mathcal{X} \setminus X$ to their current value, such that the values of variables X can be freely selected by the solver. For a solution S, the notation $n(S)$ is used to denote this neighborhood. Each neighborhood has a property which defines the amount of variables to select, found by $|n| = \alpha |\mathcal{X}|$, where $\alpha \in]0;1]$. The value of α is adaptively adjusted throughout the algorithm, which we elaborate on later. Initially it is set to $\alpha_0 = \max(1, \frac{|n|_0}{|\mathcal{X}|})$, where $|n|_0 \in \mathbb{N}^+$ is the 'target' absolute amount of variables initially. Thereby the amount of variables is selected relative to the total amount of decision variables, for better adapting to the size of each problem instance. Hence a neighborhood consists of a (small) part of the overall

solution space. A MIP solver might be able to explore this small sub-space effectively, making this iterative approach attractive. We will rely on empirical results to support this claim. One of the most prominent effects of fixing many variables is that the solver is able to greatly reduce the size of the problem in the presolve face.

The matheuristic algorithm is shown in Algorithm 1. In the remainder of this section we describe this algorithm in details. The algorithm can in principle be applied to any MIP model, however this would require generic MIP-neighborhoods. Such neighborhoods could for instance be constructed by identifying related variables in terms of amount of constraints which involves these variables. In this paper we consider the neighborhoods problem-specific, in the hope of improving performance.

Algorithm 1 Matheuristic algorithm

```
1:  input: problem instance of HSTP, neighborhoods N
2:  output: feasible solution S
3:  S := initial(HSTP)                          ▷ Construct initial solution
4:  HSTP = MIPPresolve(HSTP)                    ▷ Presolve model
5:  while stopping criteria not met do
6:      choose neighborhood n
7:      obtain variables V := n(S)
8:      fix variables X \ V to their current value
9:      invoke MIP-solver on S with timelimit t_MIP
10:     if MIP-gap ≤ g_min then
11:         increase size of n
12:     else if MIP-gap ≥ g_max then
13:         decrease size of n
14:     end if
15:     unfix all variables
16: end while
```

An initial solution is constructed in a problem-specific way in Line 3. For both our test cases, this is performed by a heuristic based on a greedy principle which we describe in detail in Section 10.4. In both cases the greedy heuristic has a low running time.

In Line 4 of the algorithm, the MIP is presolved by the solver. For most problem instances, this greatly reduces the size of the MIP. In effect, the iterative process is performed on the presolved model, which has the advantage of preventing the same presolve-steps to be performed in each iteration. Note that presolve is also performed in the iterative steps of the algorithm, and we note that it has great impact on reducing problem size when a neighborhood has been applied to the MIP model.

Lines 6-9 resides in the iterative part of the algorithm, and chooses a neighborhood, applies it to the current solution, and invokes the MIP solver.

As previously discussed, it seems advantageous to adaptively adjust the size of neighborhoods. This is done by querying the resulting gap of the MIP solver (Lines 10-14). If this gap is below a threshold $g_{\min}$, the size of the neighborhood is increased by $\alpha = \alpha + 0.02$. If it is above a threshold $g_{\max}$, the size is decreased by $\alpha = \alpha - 0.02$. Obviously, the MIP-gap depends on the applied neighborhood and its current size.

It should be remarked that the current solution S is not excluded in the search space of $n(S)$ by this approach. This has the advantage that the MIP solver is able to warm-start, as the final state of the previous solve operation is not made invalid by the added constraints.

10.3.1 Neighborhood selection

A mechanism for selecting the neighborhood to use in each iteration is needed (Line 6 of the algorithm). We use a roulette-wheel approach, where the probability of selecting neighborhood n is $P(n)$. Two factors determine the magnitude of the probability: 1) The previous improvement made by the neighborhood, relative to the total improvement made by all neighborhoods. A high value results in higher probability. 2) The total time spend in this neighborhood, relative to the total time spend by all neighborhoods. Low value results in higher probability. Associate with neighborhood n the total amount of seconds used S_n, and the total absolute improvement found by the neighborhood O_n. Let O^{total} and S^{total} denote the total absolute improvement so far and the total amount of seconds spend so far, respectively,

$$O^{\text{total}} = \sum_{n \in N} O_n \tag{10.1}$$

$$S^{\text{total}} = \sum_{n \in N} S_n \tag{10.2}$$

The following quantity is the heart of the mechanism,

$$Q_n = \begin{cases} \beta \frac{O_n}{O^{\text{total}}} + (1-\beta)\left(1 - \frac{S_n}{S^{\text{total}}}\right) & O^{\text{total}} > 0, S^{\text{total}} > 0 \\ 1 & \text{else} \end{cases} \tag{10.3}$$

where $\beta \in \mathbb{R}^+$ weights the expression. Intuitively, neighborhoods which result in large improvements and/or requires a small computational effort are favored by this expression. Notice that the condition $O^{\text{total}} > 0, S^{\text{total}} > 0$ holds for all iterations of the matheuristic except the first one. The probability for selecting neighborhood n is defined as

$$P(n) = \frac{Q_n}{\sum_{n' \in N} Q_{n'}} \tag{10.4}$$

The weighted roulette wheel mechanism for selecting neighborhoods is commonly used in ALNS algorithm, as well as in hyper-heuristics (see for instance Misir et al. (2012)).

10.4 Test Setup

Two different test cases are used to evaluate the matheuristic. Both are cases of High School Timetabling, and have the same overall goal: Schedule a set of events to timeslots, subject to the requirements of resources for each event, such that the individual timetable for each resource is feasible (obeys hard constraints), and as far as possible obeys individual requests (obeys soft constraints).

Two essential issues should be considered when selecting the subset of variables which constitutes a neighborhood $n(S)$:

- The resulting subproblem should be *solvable* by the MIP solver. Notice that it is not a requirement that the subproblem is solved to optimality, so solvable is in our terminology defined as yielding a final MIP-gap below $g_{\max}$.
- The free variables should have a structure which allows improved solutions to be found, so the free variables should be related is some way. If this is not the case, there is a good chance that the values of the fixed variables will implicitly define values for all the free variables.

Neighborhoods for both test cases are based on the same procedure, presented in Algorithm 2. The idea is to select decision variables related to the same resources, according to some way of measuring the quantity of relatedness among resources. This could for instance be the amount of events which require a specific resource. In Line 11 of the algorithm, decision-variables are selected on the basis of their dependency on the resource in question. This is done in a problem-specific way, elaborated in Section 10.4.1 and Section 10.4.2.

Algorithm 2 Neighborhood Related Resources

1: **input:** Set of problem-resources R, neighborhood size $|n|$
2: **output:** Set of decision-variables V
3: $d :=$ random element of R
4: $D := \{d\}$
5: **while** $|V| < |n|$ **do**
6: $\quad \overline{D} := R \setminus D$ ▷ Resources not processed yet
7: $\quad$ **if** $\left|\overline{D}\right| = 0$ **then**
8: $\quad\quad$ stop
9: $\quad$ **end if**
10: $\quad d :=$ best element in $\overline{D}$ according to some rank-measure
11: $\quad v :=$ decision-variables of $\mathcal{X}$ which relates to d
12: $\quad V := \{V \cup v\}$
13: $\quad D := \{D \cup d\}$
14: **end while**

10.4.1 Generalized High School Timetabling Problem (XHSTT format)

The first of our test is the generalized version of HSTP. Based on the *Extensible Markup Language* (XML) standard, the XHSTT format for modeling instances of (high) school timetabling was recently established (Post et al., 2012a). One purpose of the format is to serve as a common test-bed for researchers within the area. The *International Timetabling Competition 2011* (ITC2011) was based on instances in this format (Post et al., 2012b), and the attention level for XHSTT format is generally rising.

In Kristiansen et al. (2013) a MIP model for the XHSTT format were presented. The MIP is described in the following. A set of *events* is given, denoted $\mathcal{E}$, which should be assigned *times* (the set $\mathcal{T}$) and *resources* (the set $\mathcal{R}$). Specifically, each event contains a number of *event resources* (indexed by $er \in e$) which defines a requirement for a resource. An resource which can be assigned to event resource er is denoted by $r \in er$. Each event has a certain duration D_e, and it is possible to split events into *sub-events*, which defines a portion of the event with the same resource-requirements and with the duration D_{se}. The set of all sub-events is denoted $\mathcal{SE}$. The summed duration of all *active* sub-events of an event in a solution must equal D_e. The times are typically divided into different *time groups*, denoted by the set $\mathcal{TG}$. A timetable is sought which schedules sub-events to times and resources, such that a number of constraints is obeyed. The set of constraints is divided into soft and hard constraints, and an objective value of a solution describes the amount of violation of each of these, denoted (hardcost, softcost).

The basic decision variable of the MIP is $x_{se,t,er,r} \in \{0, 1\}$, which takes value 1 if sub-event $se \in \mathcal{SE}$ is assigned to starting-time $t \in \mathcal{T}$ and event-resource $er \in se$ is assigned resource $r \in er$, and 0 otherwise. A basic constraint of the model is (10.5), which specifies that each event-resource $er \in se$ of sub-event $se \in \mathcal{SE}$ should be assigned exactly one resource. Since it

is feasible to *not* assign a resource to an event-resource (such cases are usually penalized by the constraints of the problem), the set of resources $\mathcal{R}$ is extended with a dummy-element r_D.

$$\sum_{t\in\mathcal{T},r\in\mathcal{R}} x_{se,t,er,r} = 1 \qquad \forall se \in \mathcal{SE}, er \in se \tag{10.5}$$

The variable $y_{se,t} \in \{0,1\}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is assigned to starting-time $t \in \mathcal{T}$, and 0 otherwise. Let $|er|_{se}$ denote the amount of event resources of sub-event $se \in \mathcal{SE}$. Constraints (10.6) specifies the link to variable $y_{se,t}$ and ensures in combination with (10.5) that each sub-event is only one starting time.

$$\sum_{er\in se,r\in\mathcal{R}} x_{se,t,er,r} = |er|_{se} \cdot y_{se,t} \qquad \begin{array}{l}\forall se \in \mathcal{SE}, \\ t \in \mathcal{T}\end{array} \tag{10.6}$$

On top of these variables and constraints are build a large set of additional constraints and auxiliary variables. The full set of decision variables is $\mathcal{X} = \{x_{se,t,er,r} \cup y_{se,t}\}$.

By definition of the XHSTT format, the objective is to minimize the violation of both the hard and the soft constraints. However, solutions are first ranked on their hard cost, and secondly on their soft cost. Therefore Kristiansen et al. (2013) proposes to solve the MIP in two steps; In Step 1, a MIP with all hard constraints is build, and solved with a generic MIP solver until optimality is found or the given timelimit is reached. In case optimality is found, all soft constraints are added, as well as a constraint which ensures that the violation of the hard constraints found in Step 1 is maintained, and the MIP is again solved by a MIP solver. This is Step 2 of the procedure. In our implementation this two-step procedure is also used, with the MIP solver replaced by the matheuristic. However this has the unfortunate disadvantage that the matheuristic in general is unable to determine whether a found solution is optimal, except for the case where a solution with objective 0 is found (which is optimal for any XHSTT solution). Therefore the following heuristic stopping criteria is imposed: If in Step 1, the found solution has not been improved in the last s seconds, stop and continue to Step 2. s is set to 30% of the overall timelimit imposed for the matheuristic.

For constructing the initial solution, an algorithm based on a *greedy* principle is used. A (partial) solution S is given, as well as a set of moves $\mathcal{M}$. With each move $M \in \mathcal{M}$ is associated a value $\Delta(M,S) \in \mathbb{N}_0$ which denotes the contribution of move M to solution S. If this contribution is negative, the move results in an improved solution. The set of moves $\mathcal{M}$ consists of all possible moves $M_{se,t}$ which assigns sub-event $se \in \mathcal{SE}$ to time $t \in \mathcal{T}$, and all possible moves $M_{r,er,se}$ which assigns resource $r \in \mathcal{R}$ to event resource $er \in se$ of sub-event $se \in \mathcal{SE}$ (see also Sørensen et al. (2012)). The initial solution is constructed by selecting the move from $\mathcal{M}$ for which $\frac{\Delta(M,S)}{D_{se}}$ is most negative. The division with D_{se} is added as this has shown to produce better solutions than the pure greedy algorithm. The selected move is applied to S, all $\Delta-$ values are updated, a new move is selected, and the process is repeated until no move exists which will result in an improved solution.

10.4.1.1 Neighborhoods

Several different neighborhoods are imposed, described in the following. All are based on Algorithm 2, and hence each of them requires to define the set of resources it applies to, the rank-measure to select related resources, and a mechanism to extract decision variables for a resource.

Same Events This neighborhood applies to the set of events $\mathcal{E}$, and for each event selects the decision variables related to all sub-events of the events. The rank-measure to select a related

event is based on the amount of applicable resources which the events share. Specifically, D is a set of events, and the set $R' = \{r \in er \mid er \in e \in D\}$ is the set of applicable resources for all these events. Associate with an event $e \in \mathcal{E} \setminus D$ the number of shared resources with R', $|R' \cap \{r \in er \mid er \in e\}|$. The event for which this number is largest is selected next.

This neighborhood gives the opportunity for the algorithm to change the assigned time and resources of the sub-events of the selected events. The fact that the events are related increases the probability of achieving an improved solution.

Related Resources This applies to the set of resources $\mathcal{R}$, and selects the decision variables by finding the events for which it is possible to assign the resource to. The relatedness between resources are determined by the amount of events which the resource can be assigned to.

Continuous Times This neighborhood applies to the set of times, and selects the decision variables by means of each time. Times are selected in chronological order, such that the forthcoming time is selected next.

Time Groups This neighborhood applies to the set of time groups. For a given time group, the decision variables are selected on the basis of all times of the time group. The relatedness between time groups is determined by the amount of times which are shared among the time groups.

10.4.2 The Danish High School Timetabling Problem

The second test-case is the Danish version of the HSTP. A MIP model for this problem is established in Sørensen and Stidsen (2013a). This timetabling application is used in practice by many high schools in Denmark, and the model is *complete* in the sense that no constraints are left out in our formulation.

Let $\mathcal{E}$ denote the set of events (an event general mimics a lecture) to be scheduled. The set of resources is denoted $\mathcal{A}$. Each event requires a subset of resources (in this case classes, students and teachers) $A \subseteq \mathcal{A}$. Each event should be scheduled to one timeslot and one room, denoted by the sets $\mathcal{T}$ and $\mathcal{R}$, respectively. This is modeled by the variable $x_{e,r,t} \in \{0,1\}$ which takes value 1 if event $e \in \mathcal{E}$ is assigned to room $r \in \mathcal{R}$ and timeslot $t \in \mathcal{T}$, and 0 otherwise. In a feasible schedule, each resource and each room can only be occupied by one event in each timeslot. To ensure feasibility the sets of timeslots and rooms are both extended by a dummy-element; $\mathcal{T} = \{\mathcal{T} \cup t_D\}$, $\mathcal{R} = \{\mathcal{R} \cup r_D\}$. The objective of the problem is to minimize $\sum_{e\in\mathcal{E},r\in\mathcal{R},t\in\mathcal{T}}(\phi_{e,t} + \pi_{e,r})x_{e,r,t}$, where $\phi_{e,t}, \pi_{e,r} \in \mathbb{R}^+$ denotes the penalty for assigning event $e \in \mathcal{E}$ to timeslot $t \in \mathcal{T}$ and room $r \in \mathcal{R}$, respectively. The penalty for assigning to the dummy timeslot and the dummy room is much larger than all other penalties. Besides these requirements, many additional constraints and auxiliary variables are used in the MIP model, see Sørensen and Stidsen (2013a) for details.

Previous work on the MIP model of HSTP has shown that a *two-stage decomposition* is way more effective than solving the MIP based on the three-index variable $x_{e,r,t}$ (Sørensen and Dahms, 2014). The key idea of the decomposition is to split the model into two distinct MIPs, each based on a binary variable with two indices ($y_{e,t}$ and $z_{e,r}$, denoting time- and room-assignment, respectively). In the first MIP (*Stage I*), events are assigned to timeslots, and in the second MIP (*Stage II*) events are assigned to rooms, subject to the assigning to timeslots provided by Stage I. It is shown by Sørensen and Dahms that this decomposition maintains near-optimality of the original MIP, and that the decomposition obtains both better solutions and better lower bounds. Furthermore, it is argued that future research should focus on solving the Stage I model, as the

Algorithm 3 Greedy algorithm

```
1: input: problem instance of HSTP
2: S := empty solution
3: loop
4:     M*_{e,t} := min_{e∈E,t∈T} Δ(M_{e,t}, S)
5:     if Δ(M*_{e,t}, S) > 0 then
6:         return S
7:     end if
8:     apply M*_{e,t} to S
9: end loop
```

Stage II model is easy to solve using a state-of-the-art MIP solver. Therefore the matheuristic presented in this paper solves the Stage I model, and we rely on a MIP solver to handle Stage II.

For constructing the initial solution, a basic greedy algorithm is used, see Algorithm 3. The move $M_{e,t}$ denotes the assigning of event $e \in \mathcal{E}$ to timeslot $t \in \mathcal{T}$. The algorithm iteratively finds the move which lowers the objective value the most, and applies it to the solution.

10.4.2.1 Neighborhoods

The following neighborhoods are used.

Continuous Timeslots This neighborhood starts by selecting a random timeslot, and selects decision variables related to this timeslot. The next timeslot selected is the one immediately following the previously selected timeslot.

Related Classes This neighborhood applies to the set of classes, and selects decision variables based on the events related to each class. The relatedness between classes is determined by the amount of events which are shared among the classes. A variant of this neighborhood is denoted *Related Classes Sparse* uses the same criteria, but selects only $\frac{1}{10}$ of the decision variables of the class in question in each iteration of the selection process. This variant of the neighborhood allows the set of selected decision variables to be more loosely coupled w.r.t. classes.

Related Students/Teachers This neighborhood is similar to *Related Classes* except that it applies to the sets of students and teachers instead of the set of classes. Also this neighborhood is used in a sparse variant.

10.4.3 Parameter values

The described matheuristic contains a number of parameters which should be tuned for optimal performance. Automated tuning procedures for heuristics have started to emerge (Birattari (2005); Hutter et al. (2009), but these are still time demanding algorithms which provide no guarantee to find the optimal set of parameter values. Therefore the tuning procedure has been done ad-hoc, using a set of instances for each of the test-problems which are distinct from the set of instances used to establish computational results. Table 10.1 shows each parameter and its corresponding chosen value.

Table 10.1: Parameter values

Sym.	Description	Range	Value
$\|n\|_0$	Neighborhoods initial size	$\mathbb{Z}^+$	1000
β	Neighborhood weight factor	$[0, 1]$	0.70
t_{MIP}	Timelimit for MIP solver	$\mathbb{R}^+$	5
$g_{\min}$	If MIP-gap below this, increase the size of the neighborhood	$[0, 100]$	2
$g_{\max}$	If MIP-gap above this, decrease the size of the neighborhood	$[0, 100]$	10

10.5 Computational Results

All tests were ran on a 64-bit Windows machine with an Intel i7 930 CPU @ 2.80GHz and 12 GB RAM. Gurobi 5.5.0 was used as MIP solver, which is among the best MIP solvers available according to recent benchmarks (Mittelmann, 2013). For all tests, the Gurobi pseudo-parameter `MIPFocus` was set to value 1, emphasizing that we are interested in feasibility. Furthermore, Gurobi was only allowed to use a single CPU thread.

10.5.1 XHSTT - International Timetabling Competition 2011

To evaluate the matheuristic on the XHSTT format, the collection of datasets used in Round 2 of ITC2011 is used. In this round of the competition, the finalist algorithms were tested on 18 previously unseen datasets on the same machine. The organizers of the competition provided a tool which can be used to benchmark other machines to achieve the equivalent runtime (772 seconds in our case), facilitating a fair comparison. In the following we compare our algorithm with the finalists of ITC2011, which is in fact a comparison on even terms, except for the fact that commercial software was not allowed in ITC2011 (which conflicts with our use of Gurobi). Future research will show how the matheuristic will perform with a non-commercial MIP solver. As the matheuristic is stochastic, it was ran 5 times on each dataset.

An objective of a XHSTT solution is written as (hard cost, soft cost), denoting the violation of the hard constraints and the soft constraints, respectively. In case the hard cost has value 0, only the soft cost is written. Table 10.2 shows the obtained results. The table shows that the matheuristic is among the best algorithms, obtaining the best result on 10 datasets. The table also shows the average rank of each solution method computed in the following way: For each dataset, rank each solution from 1 to 6, 1 being the best. The row *Avg. Ranks* shows the average of all ranks for each solution method. According to this measure, the matheuristic performs second best.

Table 10.2: Performance of the MIP using same running time as specified in ITC2011. For each instance is listed the average solution found from each of the competitors of ITC2011, and the solution obtained by the MIP formulations. Columns *GOAL*, *HySTT*, *Lectio* and *HFT* denote the finalists of ITC2011. Column *MIP* denotes the performance of the MIP of Kristiansen et al. (2013). For the matheuristic, column *Obj* denotes the average of the found objectives, column σ denotes the standard deviation of the found objectives, and columns T_1 and T_2 denotes the running time for Step 1 and Step 2, respectively. The best solutions are marked in **bold**. Row *Avg. Ranks* denotes the average ranking of each solution method, 1 being best.

							Matheuristic			
		GOAL	HySST	Lectio	HFT	MIP	Obj	σ	T_1	T_2
BR	Instance2	(1, 62)	(1, 77)	38	(6, 190)	46	**6**	(0, 1)	4	768
BR	Instance3	124	118	152	(30, 283)	39	**27**	(0, 2)	9	763
BR	Instance4	(17, 98)	(4, 231)	(2, 199)	(67, 237)	(5, 286)	**58**	(0, 3)	51	721
BR	Instance6	(4, 227)	(3, 269)	230	(23, 390)	682	**57**	(0, 4)	11	761
FI	ElementarySchool	4	(1, 4)	**3**	(30, 73)	**3**	**3**	(0, 0)	18	754
FI	SecondarySchool2	**1**	23	34	(31, 1628)	(1604, 3878)	6	(0, 3)	32	740
GR	Aigio	**13**	(2, 470)	1062	(50, 3165)	(1074, 3573)	180	(1, 104)	410	362
IT	Instance4	454	6926	651	(263, 6379)	17842	**48**	(0, 7)	71	701
XK[1]	Instance1	(59, 9864)	(1103, 14890)	(275, 7141)	(989, 39670)	(3626, 2620)	**(9, 23525)**	(10, 2359)	772	-
NL	Kottenpark2003	**90928**	(1, 56462)	(50, 69773)	(209, 84115)	(8491, 6920)	(238, 43143)	(0, 0)	772	-
NL	Kottenpark2005A	**(31, 32108)**	(32, 30445)	(350, 91566)	(403, 46373)	(2567, 53)	(566, 19968)	(0, 0)	772	-
NL	Kottenpark2008	**(13, 33111)**	(141, 89350)	(209, 98663)	-	(14727, 5492)	(6112, 353671)	(2465, 22992)	772	-
NL	Kottenpark2009	**(28, 12032)**	(38, 93269)	(128, 93634)	(345, 99999)	(17512, 140)	(9418, 705605)	(0, 0)	772	-
ZA	Woodlands2009	(2, 14)	(2, 70)	**(1, 107)**	(62, 338)	(1801, 705)	(2, 429)	(2, 355)	645	127
ES	SpainSchool	894	1668	2720	(65, 13653)	(1454, 11020)	**485**	(1, 52)	163	609
GR	WesternGreece3	**6**	11	(30, 2)	(15, 190)	25	**6**	(0, 1)	7	765
GR	WesternGreece4	**7**	21	(36, 95)	(237, 281)	81	12	(0, 3)	23	749
GR	WesternGreece5	**0**	4	(4, 19)	(11, 158)	15	**0**	(0, 0)	6	66
Avg. Ranks		2.2	3.4	3.3	5.3	4.6	2.3			

[1] Kosova.

10.5.2 The Danish High School Timetabling Problem

To evaluate the developed matheuristic on the Danish case of HSTP, it is tested against 100 real-life datasets taken from the database of the commercial system Lectio[1]. These datasets are the same ones used in Sørensen and Dahms (2014) and Sørensen and Stidsen (2013a), allowing for a comprehensive comparison with other solution approaches. The previous solution approaches are the following:

- *Adaptive Large Neighborhood Search heuristic*: In Sørensen and Stidsen (2013a) a heuristic based on ALNS is presented.
- *3-index MIP model*: This approach solves the MIP model based on the 3-indexed variable $x_{e,r,t}$ using Gurobi 5.0.1 with standard settings. The MIP is presented in Sørensen and Stidsen (2013a).
- *Two-Stage Decomposition (TSD)*: This denotes the MIP decomposition method described in Section 10.4. We compare with two versions of TSD, with and without room penalties not integrated in Stage I, denoted TSD and $\text{TSD}^{\text{RoomLB}}$, respectively. See Sørensen and Dahms (2014) for more details of this approach.

The timelimit was set to 7200 seconds, but we report the found solution at several different points in time. In fact, the timelimit for all solution approaches are 7200s, except for the ALNS algorithm which had a 240s timelimit. To compare with the ALNS heuristic, we log the solution found at 240s. The matheuristic was ran 5 times on each dataset, and the numbers reported are averages. All tests of the previous solution approaches were run on the same type of machines as used for our tests.

Since we use the matheuristic to solve the Stage I model, as described in Section 10.4, the Stage II model must be solved to obtain a solution to the overall problem. In Sørensen and Dahms (2014) it was shown that the Stage II model in general is easy to solve by a MIP solver. Therefore we impose a timelimit of 60s to the Stage II model. These 60s are deduced from the overall timelimit, e.g. the objective reported for 240s is the result of 180s seconds for the Stage I model with the matheuristic, and 60s for the Stage II model using Gurobi. Since a fixed timelimit is imposed for the two stages, it might happen that Gurobi were unable to find a feasible solution. In such cases we report the objective value as "-". The same notation is used to denote that a lower bound is not known for a given dataset.

Table 10.3 shows the obtained results, and Table 10.4 summarizes some key numbers. For a timelimit of 240s, the matheuristic is compared with the ALNS algorithm. This shows that the ALNS algorithm performs best on 71 datasets, and the average gap across all datasets is 19.53%. This outperforms the matheuristic, which is best on 29 datasets, and has an overall gap of 23.38%. For a timelimit of 7200s, the matheuristic outperforms the 3-index MIP and both versions of TSD, with an average gap of 15.20% and the best average solution on 81 datasets.

[1] Cloud-based ERP system for high schools used by the majority of high schools in Denmark, http://www.lectio.dk

Table 10.3: Computational results for the Danish High School Timetabling Problem. For each dataset is shown the objective obtained by each solution method. Column *LB* denotes the best known lower bound. For the matheuristic is shown the value of the initial solution *Greedy*, as well as the obtained average objective value at different points in time. For each datasets and for each solution method, the gap to the best known lower bound is calculated by $\frac{\text{obj}-LB}{\text{obj}}$, and the last row *Avg. Gap* denotes the average gap across all datasets.

	Previous methods					Matheuristic					
Dataset	ALNS	3-idx MIP	TSD	$\text{TSD}^{\text{RoomLB}}$	LB	Greedy	120s	240s	360s	480s	7200s
AalborTG2012	6317	6118	6018	6005	5946	8218	6831	6161	6081	6055	6035
AarhusA2011	10037	58015	15872	18122	5986	14276	12900	11195	10610	10202	9236
AarhusA2012	7971	17096	8947	11936	6005	11888	9854	8204	7579	7324	7040
Aars2009	14900	49504	20780	24240	12641	22864	20495	16588	15468	14744	14043
Aars2010	16268	81970	25057	24692	14151	22291	20914	18139	17314	16681	15742
Aars2011	14256	77967	30623	33790	10501	18164	17216	15710	14978	14400	13442
Aars2012	10701	55049	21206	20274	8044	16956	15096	12626	11572	11088	10094
Alssund2010	9967	52717	23173	21455	6876	18885	16075	12962	11853	11044	9325
Alssund2012	29803	108810	108810	108810	-	42831	42815	42492	41995	40846	32137
BagsvaG2010	3960	6777	3916	4051	3227	6164	4105	3740	3643	3607	3575
BirkerG2011	42063	119600	119600	119600	-	62242	62225	62067	61581	60174	46790
BirkerG2012	19552	110180	19322	18182	17709	19993	19209	18806	18737	18711	18609
BjerrG2009	16877	52639	35514	27396	12288	26422	24440	20900	18799	17287	15546
BjerrG2010	4983	12868	5788	5977	3928	7737	5930	5014	4837	4678	4500
BjerrG2011	6334	13009	9302	11676	4142	10355	9122	7581	7007	6464	5600
BjerrG2012	8023	17200	15265	17404	5055	13001	11948	9768	9016	8403	7322
BroendG2012	2040	2005	1929	1928	1881	3284	1946	1934	1933	1933	1932
CPHWGym2010	6775	34415	19363	16589	3759	11345	10116	8909	8147	7482	6239
CPHWGym2011	5679	38232	16212	15046	4103	11779	9477	7051	6332	5759	5203
CPHWGym2012	6762	40945	15543	17194	4215	12311	11159	9668	8954	7948	6343
CPHWHG2012	11077	46625	23088	23219	8338	20253	18874	15494	13202	11930	10385
CPHWHTX2010	11342	27174	15943	19314	9259	14277	13383	12182	11774	11466	10742
CPHWHTX2011	20734	22466	20708	20632	20470	22619	20898	20690	20657	20646	20646
CPHWHTX2012	16256	25998	21392	22481	14531	19035	17924	16810	16440	16235	15789
DetFG2012	7560	8017	7265	7258	7168	10123	7940	7387	7331	7325	7313
DetKG2010	2947	6058	4006	4102	1821	6113	4253	3196	2941	2847	2661
DetKG2011	2820	5594	4366	4577	1781	5831	4234	3196	2935	2732	2460
EUCN2009	3737	7557	4298	5001	2982	5536	4649	3895	3716	3583	3395
EUCN2010	3882	4231	3463	3430	3375	5065	3897	3634	3535	3529	3500
EUCN2011	1468	1435	1430	1426	1395	1681	1472	1456	1450	1443	1433
EUCN2012	3289	9430	5059	5913	2363	6138	5038	3621	3429	3365	2972
EUCNHG2010	1505	1476	1421	1408	1378	2014	1459	1411	1408	1407	1405
EUCS2012	3714	4689	3783	3695	3584	4366	3695	3666	3663	3661	3659
FaaborgG2008	68124	125330	125330	125330	-	-	-	-	-	-	-
FalkonG2009	10449	88890	88890	88890	-	19562	19628	19118	18135	15912	10403
FalkonG2011	8584	76170	16543	20758	5183	15380	14166	11745	10315	9435	7688
FalkonG2012	10143	100190	16666	14908	6105	18664	16205	12898	11631	10645	9302
GUAasia2010	6527	6579	6461	6422	6374	6567	6430	6431	6431	6431	6431
GUQaqor2011	6674	19623	10005	11396	4554	10612	9383	8256	7588	7142	6444
GUQaqor2012	5733	11488	7619	9650	4324	9645	7754	6397	6004	5735	5492
HadersK2011	7128	51190	14229	16494	3909	14268	12790	10628	9467	8444	6851
HasserG2010	11963	96790	96790	96790	-	22438	22480	21863	21109	19543	11814
HasserG2011	16061	99840	99840	99840	-	27370	27510	27028	26555	25307	16744
HasserG2012	18338	112160	112034	112160	-	28125	28231	27344	26004	24316	17223
HerningG2010	37	37	37	37	37	35	35	35	35	35	35
HerningG2011	15091	163785	23117	26410	9829	20432	17329	14437	13909	13526	12752
HerningG2012	13147	185433	14952	19834	9817	-	-	-	-	-	-
HoejeTaG2008	2958	6292	2707	2775	2587	4092	3434	2949	2849	2801	2742
HoejeTaG2009	9157	45260	26066	27779	5773	16478	15985	14665	13329	11872	9362
HoejeTaG2010	9862	45095	25678	27886	6188	16538	15542	13862	12609	11514	9524

Continued on next page

Table 10.3 – continued from previous page

	Previous methods					Matheuristic					
Dataset	ALNS	3-idx MIP	TSD	TSD$^{\text{RoomLB}}$	LB	Greedy	120s	240s	360s	480s	7200s
HoejeTaG2011	10158	51050	32630	30327	6726	15428	14844	13945	13379	12497	9894
HoejeTaG2012	12502	72455	18627	39326	7952	17056	15502	13375	12008	11219	9739
HorsenS2009	3111	3100	3100	3100	3100	3186	3100	3100	3100	3100	3100
HorsenS2012	10056	86090	86090	86090	-	20756	20611	19831	18956	17605	10497
Johann2012	23001	92575	27781	29491	19590	27947	27300	25508	24354	23338	22061
KalundG2011	38479	126150	126150	126150	-	48260	48327	48130	47696	46977	41105
KalundG2012	26768	123010	123010	123010	-	38923	39098	38948	38574	37634	31708
KalundHG2010	5631	12103	6351	6605	4642	8167	6971	6112	5809	5521	5254
KoebenPG2012	888	1872	874	1052	645	1441	944	845	825	814	790
KoegeH2012	11418	108347	20150	20390	9440	16301	12720	11298	10942	10714	10454
KongshoG2010	4296	8889	7954	7208	2488	8983	6342	4850	4391	4122	3635
MariageG2009	8013	54030	20138	17506	5286	14176	12168	9691	8929	8324	7385
MorsoeG2012	5651	42762	10241	11674	3947	7250	6863	6289	6151	5870	5418
NaerumG2008	24104	118370	117894	117894	-	43055	42047	37086	32760	29054	23251
NaerumG2009	7667	100450	6681	5466	5114	8614	6144	5676	5647	5630	5636
NielsSG2011	4953	10464	6132	5397	3412	8381	7106	5637	5077	4784	4444
NielsSG2012	6952	12747	8003	9192	5738	9367	7587	6761	6619	6548	6403
NordfynG2012	5160	8201	4890	5510	4152	6410	5239	4895	4803	4760	4705
NyborgG2011	13944	94059	31809	85816	6129	22029	21902	20815	19686	18161	13741
OdderCfU2010	18219	59540	40032	38875	12865	26010	25912	25726	25317	24679	20016
OdderG2009	9308	59851	57586	24686	5361	15776	15328	13731	12793	11456	8796
OdderG2012	12307	17402	14888	27199	9688	16982	14061	12241	11909	11651	11310
OrdrupG2010	13663	75700	12936	18101	10810	22742	18723	14628	13238	12362	11877
OrdrupG2011	21612	116400	31329	28884	17692	-	-	-	-	-	-
RibeK2011	21679	61945	43175	39107	18055	27405	26894	25786	25053	24353	22067
RysenG2010	39971	110690	110690	110690	-	46067	44259	41892	41447	41202	40066
RysenG2011	22260	100313	25989	68927	19725	29641	26402	22759	22239	22023	21549
RysenG2012	19841	110111	22156	59124	16708	24265	22655	20876	19926	19646	19169
SanktAG2012	4207	4624	3911	3721	3538	4607	3931	3760	3746	3729	3723
SkanderG2010	7209	7708	6875	6485	6238	7459	7026	6644	6587	6556	6540
SkanderG2011	22525	88470	88470	88470	-	32962	32576	31306	30276	28752	22546
SkanderG2012	20138	98487	95319	95319	-	35642	35716	35603	35118	34178	25121
SkiveG2010	43120	194740	194740	194740	-	-	-	-	-	-	-
SlagelG2012	32167	162960	162765	162960	-	-	-	-	-	-	-
SoendS2011	11776	83560	83560	83560	-	22600	21286	17996	15979	14008	10832
SoendS2012	8420	17778	11915	24668	6838	11170	76062	42608	74114	89229	88714
StruerS2012	73361	-	207488	211960	-	-	-	-	-	-	-
VardeG2012	10777	20933	20622	20496	5921	15714	14075	11739	10941	10098	8644
VejenG2009	11264	69450	69450	27954	7290	23373	23874	20421	41357	47520	19884
Vejlefjo2011	13514	52035	18043	22066	8805	18934	18587	17760	17173	16020	13037
VestfynG2009	5973	11606	5999	5032	4211	8423	5701	4887	4738	4678	4631
VestfynG2010	6761	16895	5974	5239	4308	10313	7403	5940	5703	5430	5016
VestfynG2011	7013	13624	6657	6522	5159	10857	7487	6397	6217	6149	6072
VestfynG2012	5244	11095	5212	5319	4315	8207	5435	4958	4905	4848	4762
ViborgK2011	14923	99170	99170	99170	-	24813	25041	24683	24113	22923	17081
ViborgTG2009	10216	19891	12077	13387	8740	12950	11493	10284	9967	9803	9595
ViborgTG2010	4932	12727	10226	10665	4146	6904	5796	5090	4927	4842	4664
ViborgTG2011	7478	16433	9808	11088	6772	9596	8104	7617	7468	7383	7276
VirumG2012	27738	140883	32183	79111	19486	25321	24387	23465	23134	22825	22197
VordingbG2009	8568	17025	9905	8972	5787	12390	10188	8560	8109	7768	7386
Avg. Gap	19.53	53.01	33.57	36.68		36.03	29.41	23.38	21.26	19.34	15.20

	ALNS	3-idx MIP	TSD	TSD^{RoomLB}	Matheuristic
Best@240s	71				29
Best@7200s		3	8	15	81

Table 10.4: Results summarized for the Danish High School Timetabling Problem. The number of best found solutions at different points in time (including draws).

10.6 Conclusion

This paper has shown a matheuristic which can be applied to general MIPs, but which uses problem-specific knowledge to explore small portions of the solution space iteratively. The algorithm has been applied to two different cases of high school timetabling: The generalized problem based on the XHSTT format, and the Danish version of the problem.

Extensive computational results were established for both versions of the problem, and good performance was shown in both cases. In case of the generalized problem, the algorithm were shown to almost perform as well as the winner of round 2 of the International Timetabling Competition 20111. In case of the Danish version of the problem, the matheuristic is the best known algorithm given a timelimit of 7200s, with an average gap of 15.20% to the best known lower bound. In case of a timelimit of 240s, the matheuristic is outperformed by another heuristic. These results show the potential of the hybridization of Metaheuristics and Mathematical Programming.

In terms of future research, we mention that the matheuristic could be a general solution algorithm for MIPs if it is equipped with general-purpose neighborhoods. Such neighborhoods could for instance select decision variables based on the amount of constraints which applies to these variables. Furthermore, the computational overhead for the algorithm on very large MIPs is still significant. An improved algorithm could possibly handle this by not building the entire MIP initially.

Bibliography

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151, 2011. ISSN 1568-4946.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003. ISSN 0025-5610.

F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36:267–306, September 2009. ISSN 1076-9757.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Integer programming for the generalized (high) school timetabling problem. *Journal of Scheduling*, Submitted 5/9-2013, 2013.

V. Maniezzo, T. Stützle, and S. Voß. Matheuristics: Hybridizing metaheuristics and mathematical programming. *Annals of Information Systems*, 10, 2009a.

V. Maniezzo, S. Voss, and P. Hansen. Special issue on mathematical contributions to metaheuristics. *Journal of Heuristics*, 15(3), June 2009b.

M. Misir, K. Verbeeck, P. Causmaecker, and G. Berghe. An intelligent hyper-heuristic framework for chesc 2011. In Y. Hamadi and M. Schoenauer, editors, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 461–466. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-34412-1.

H. Mittelmann. Benchmarks for optimization software. `http://plato.asu.edu/bench.html` [Accessed 9/9-2013], 2013.

N. Pillay. A survey of school timetabling research. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

M. Prandtstetter and G. R. Raidl. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *European Journal of Operational Research*, 191(3):1004 – 1022, 2008. ISSN 0377-2217.

D. Ryan. It is time to enjoy the best of both worlds. In *The 46th ORSNZ Conference*, Victoria University of Wellington, New Zealand, 10-11 December 2012.

M. Sniedovich and S. Voß. The corridor method: a dynamic programming inspired metaheuristic. *Control and Cybernetics*, 35(3):551, 2006.

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

M. Sørensen and T. Stidsen. Comparing solution approaches for a complete model of high school timetabling. Technical Report 5.2013, DTU Management Engineering, Technical University of Denmark, March 2013a.

M. Sørensen and T. R. Stidsen. Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. *Annals of Operations Research*, PATAT 2012 SI:Submitted Jan 21. 2013, 2013b.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

E. D. Taillard and S. Voss. Popmusic — partial optimization metaheuristic under special intensification conditions. In *Essays and Surveys in Metaheuristics*, volume 15 of *Operations Research/Computer Science Interfaces Series*, pages 613–629. Springer US, 2002. ISBN 978-1-4613-5588-5.

S. Voss. Meta-heuristics: The state of the art. In A. Nareyek, editor, *Local Search for Planning and Scheduling*, volume 2148 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42898-5.

Chapter 11 Paper F

The Consultation Timetabling Problem at Danish High Schools

Simon Kristiansen[1,2], Matias Sørensen[1,2], Michael B. Herold[2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract In the different stages of the educational system, the demand for efficient planning is increasing. This article treats the $\mathcal{NP}$-hard *Consultation Timetabling Problem*, a recurrent planning problem for the high schools in Denmark, which has not been described in the literature before. Two versions of the problem are considered, the *Parental Consultation Timetabling Problem* (PCTP) and the *Supervisor Consultation Timetabling Problem* (SCTP). It is shown that both problems can be modeled using the same Integer Programming model. Solutions are found using the state-of-the-art MIP solver Gurobi and *Adaptive Large Neighborhood Search* (ALNS), and computational results are established using 300 real-life datasets. These tests show that the developed ALNS algorithm is significantly outperforming both Gurobi and a currently applied heuristic for the PCTP. For both the PCTP and the SCTP, it is shown that the ALNS algorithm in average provides results within 5% of optimum. The developed algorithm has been implemented in the commercial product Lectio, and is therefore available for approximately 95% of the Danish high schools.

Keywords High School • Timetabling • Metaheuristics • Integer Programming • Adaptive Large Neighborhood Search • F-Race Tuning

11.1 Introduction

The *Consultation Timetabling Problem* (CTP) is a recurrent planning problem for the high schools in Denmark, which concerns the creation of a schedule for student-teacher meetings, given the students requests of teachers, subject to various soft constraints and resource constraints. The problem has not been described in the literature before, but it shares some properties with other problems within the educational sector, see Section 11.2.1. There exists several variants of the problem. In this paper we consider the two most important versions for the Danish high schools, namely the *Parental Consultation Timetabling Problem* (PCTP) and the *Supervisor Consultation Timetabling Problem* (SCTP).

This paper is written in cooperation with the Danish company MaCom A/S. MaCom A/S is the developer of the cloud-based high school administration system Lectio, which handles all sorts of administrative tasks for the high schools, including a GUI and a heuristic-based solver for

the PCTP. Through this cooperation we have access to real-life data for approximately 95% of all Danish high schools, which constitutes thousands of datasets. We will provide computational results for 300 of these datasets, which is a very big amount of real-life data compared to the majority of scheduling literature.

Our task of this paper is to give a detailed description of the CTP, and model it using Integer Programming. This model should support both the PCTP and the SCTP. To find solutions, both a state-of-the-art MIP solver and an *Adaptive Large Neighborhood Search* (ALNS) heuristic is attempted. These solution approaches are compared to the existing heuristic used in Lectio, and the best approach is made available for all users of Lectio.

11.2 Consultation Timetabling Problem

In the following we describe the CTP in details, starting with specifications of the two versions of the problem.

Parental Consultation Timetabling Problem: Once or twice a year the high schools invite the students and their parents to participate in meetings with the teachers of the student. The goal of these meetings is to allow the teachers to inform of the educational progress of the student, and possibly address relevant problems. Parental consultations usually take place in the evening of a normal work day, and each meeting generally has a duration between 5 and 15 minutes. The order of events for parental consultations is the following: The high school administration selects days where the meetings should take place, and for each day a feasible time-interval is chosen. Each student (usually in collaboration with his parents) makes prioritized requests of groups of teachers he would like to meet. Few of these teacher groups contain more than a single teacher, because the student is taught by only one teacher in most classes. Usually the high school also allows the students to request specific time intervals, within the overall time interval on each day, where the student will be available for meeting teachers. Given the student's choice of teachers and time intervals, it is then up to the high school administration to decide which teachers a student should meet, and when the meetings should take place.

Supervisor Consultation Timetabling Problem: In the last year of a high school education, the students are required to make a large study project (Danish: *Studieretnings Projekt*). Each student selects two subjects/courses as combined subject for his project, e.g. English and History. Each student is then assigned two teachers whom will be his supervisors for the project. The objective of the SCTP is to plan meetings between the students and their respective supervisors. The goal of these meetings is for the supervisors to provide the student with some useful hints for problem definition, literature research, etc. Typically supervisor consultations take place in the daytime, where both the student and the corresponding teachers are located at the high school.

From a timetabling point of view, these two types of consultations are almost identical. For both types, as many as possible of the meeting requests should be fulfilled, and both essentially contain the same constraints. Therefore we will in this paper model both types of consultations using the same Integer Programming model. In the remainder of this paper we refrain, as much as possible, from distinguishing between the two variants of the problem, and will by CTP denote the problem which is the superset of the PCTP and the SCTP.

In the following further details of the soft constraints of the CTP is given. These soft constraints define various scheduling preferences for the students and teachers.

A contiguous streak of meetings for a teacher or student are from now on denoted a *sequence*. A time slot is *void* for a teacher or student if the time slot is empty and no meetings are scheduled in either earlier time slots or in later time slots. A *break* for a student or teacher is defined as a time slot which is not void, and which has no meetings assigned. Void time slots must be distinguished from breaks because they do not effect the density of a schedule. This is due to the fact that students and teachers are not obligated to stay at the school throughout the entire duration of the consultation. Given these definitions, we formulate the following soft constraints:

- It is attempted to achieve a solution where the positions of the granted meetings for a given individual are placed in a sequence. I.e. for both students and teachers the number of breaks should be minimized. This is to achieve a schedule with as little waiting time as possible. However, for the students it is possible for the high school administration to declare whether they need a break after each meeting. This is usually selected if there exists "traveling" time between the meeting rooms where the teachers are located. The CTP only takes consultation meetings into consideration when determining a sequence, and not other activities.

- When assigning a meeting to a time slot, the availability of the student and teacher must be taken into consideration. The high school administration decides whether this constraint should be defined as a hard- or a soft-constraint. It is common that in case of SCTP, this is defined as a soft-constraint as it is feasible for the students to leave classes to have meetings with their supervisors. In case of the PCTP, this constraint is usually treated as a hard-constraint, as a solution should respect activities such as meetings, study-trips, etc.

- It is undesirable for teachers to have too long sequences of meetings. Therefore a maximum on the length of sequences for teachers is given (treated as a soft-constraint). This is not required for the students, since they typically have a low number of requests.

- For a consultation which spans over several days it is desired that a student or a teacher only have meetings in one of the days, such that they are not obligated to attend both. This is especially critical for students, as they have a low number of requests.

- The high school administration prefers if the meetings are placed as close as possible to a specific time slot on each day. This is usually selected as the first timeslot on each day.

Figure 11.1 shows an example of a consultation schedule on one day. The schedule contains 1 void time slot, 2 breaks, and 7 consultations.

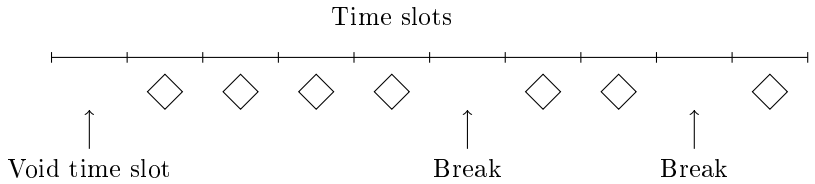


Figure 11.1: Example of a feasible consultation schedule.

11.2.1 Literature Review

The CTP has, to the best of our knowledge, not been described in the literature before. However there exist many types of related timetabling problems within the educational system, see Schaerf (1999); Burke and Petrovic (2002); McCollum (2006); Pillay (2010) for overviews of this field. Problems such as *Course Timetabling* and *Student Sectioning* have been looked in to, e.g. Tripathy (1984); Erben and Keppler (1996); Carter and Laporte (1998); Müller and Murray (2010). Related problems for Danish high schools include Kristiansen et al. (2011) and Kristiansen and Stidsen (2012) regarding the *Elective Course Planning Problem*, and Sørensen and Stidsen (2012) regarding the timetabling problem. For all these problems it applies that they attempt to assign requests to time slots in a given schedule.

The requirement for compact schedules is well known in educational timetabling. In Santos et al. (2012) the Class-Teacher Timetabling Problem with Compactness Constraints is described. The compactness is defined in terms of teacher "holes", which is equivalent to our definition of breaks. The teacher holes are modeled with a linear IP, which entails the need for two auxiliary variables, and three additional constraints. This approach can be directly applied to the CTP. de Haan et al. (2007) specifies that the timetabling problem at Dutch high schools requires compact schedules for both classes and teachers. This is addressed using heuristic methods. For Greek high schools the situation is similar, see e.g. Birbas et al. (2009). Here the problem is handled using a MIP solver for a complicated IP.

11.3 Integer Programming model

The following IP model for the CTP aims at maximizing the number of granted meeting requests and minimizing the violating of soft constraints, while respecting the hard constraints.

A consultation problem at a high school contains set of students S, a set of teachers T, a set of teacher groups G, an ordered set of time slots B and a set of of days D. $V_{b,d} \in \{0,1\}$ takes value 1 if time slot b is part of day d, and zero otherwise.

The decision whether a student s is given a meeting with teacher group g in time slot b is defined by the binary variable $x_{s,g,b} \in \{0,1\}$. The profit of meeting (s,g) in timeslot b is given by $\alpha_{s,g,b} \in \mathbb{R}^+$. The basic objective function is therefore

$$\max \sum_{s,g,b} \alpha_{s,g,b} x_{s,g,b} \tag{11.1}$$

11.3.1 Unavailabilities

In some situations it can be allowed to interrupt other activities at the high schools to satisfy a meeting request. Let $D_{t,b} \in \{0,1\}$ take value 1 if teacher t is not available (i.e. occupied by other activities) in time slot b, and zero otherwise. Let $E_{s,b} \in \{0,1\}$ be the completely analogous parameter for the students. If a consultation meeting is placed in a time slot where either a student or a teacher has some other activities, it is penalized by the following.

$$-\sum_{s,g,b} \left(\sum_{t} \delta_t \cdot D_{t,b} \cdot P_{g,t} + \delta_s \cdot E_{s,b} \right) x_{s,g,b} \tag{11.2}$$

If it is not allowed to interrupt activities this term is not added to the objective, and these constraints take the form of hard-constraints by forbidding meetings of teacher t in time slot b if $D_{t,b} = 1$, and likewise for student s in time slot b if $E_{s,b} = 1$.

It is of course not allowed to assign a student to a consultation meeting with a teacher group, if the student has not requested this teacher group. And is it not allowed for the student or teacher to have more than one meeting in each time slot. This imposed the following constraints. The parameter $P_{t,g} \in \{0,1\}$ takes value 1 if teacher t is in teacher group g, and zero otherwise. $R_{s,g} \in \{0,1\}$ takes value 1 if student s has requested teacher group g, and zero otherwise. $C_{s,b} \in \{0,1\}$ takes value 1 if student s has requested time slot b.

$$\sum_{b} x_{s,g,b} \leq R_{s,g} \qquad \forall\, s, g \tag{11.3}$$

$$\sum_{g} x_{s,g,b} \leq C_{s,b} \qquad \forall\, s, b \tag{11.4}$$

$$\sum_{s,g} P_{g,t} \cdot x_{s,g,b} \leq 1 \qquad \forall\, t, b \tag{11.5}$$

11.3.2 Undesirable breaks

One of the undesirable properties of the CTP is the breaks for both students and teachers. Let the variables $z_{s,d} \in \mathbb{N}$ and $w_{t,d} \in \mathbb{N}$ be the number of breaks in day d for a student and for a teacher, respectively. As we do not penalize void time slots as shown in Figure 11.1, we need to know when a individual have his first and last meeting. Let variables $f^{\text{first}}_{s,d}$ and $f^{\text{last}}_{s,d}$ denote the timeslot of the first and last meeting for student s, respectively. Let $h^{\text{first}}_{t,d}$ and $h^{\text{last}}_{t,d}$ be the analogous variables for teacher t. Let variable $n_{s,b,d} \in \{0,1\}$ take value 1 if student s is placed in time slot b on day d. The idle time slots for a student is then given by the following constraints

$$\sum_{g} V_{b,d} \cdot x_{s,g,b} = n_{s,b,d} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{11.6}$$

$$f^{\text{last}}_{s,d} - f^{\text{first}}_{s,d} + 1 - \sum_{b} n_{s,b,d} \cdot (1 + HS) + HS = z_{s,d} \qquad \forall\, s, d \tag{11.7}$$

$$|B|_d - (|B|_d - \text{ord}(b)) \cdot n_{s,b,d} \geq f^{\text{first}}_{s,d} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{11.8}$$

$$\text{ord}(b) \cdot n_{s,b,d} \leq f^{\text{last}}_{s,d} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{11.9}$$

The parameter $HS \in \{0,1\}$ indicates whether a student is required a break after each meeting, zero otherwise. This parameter is selectable for the user of Lectio. We want to penalize the cost such that it increases exponential on the number of breaks. The cost function is modeled as a piece-wise linear function by introducing a new variable $vs_{s,d,j} \in \{0,1\}$, where $j \in 1, ..., m$, which takes value 1 if student s has j breaks in day d. This imposes the following constraints.

$$\sum_{j} vs_{s,d,j} \cdot \text{ord}(j) = z_{s,d} \qquad \forall\, s, d \tag{11.10}$$

$$\sum_{j} vs_{s,d,j} = 1 \qquad \forall\, s, d \tag{11.11}$$

As for the teacher let variable $p_{t,b,d} \in \{0,1\}$ take value 1 if teacher t has a meeting in time slot b on day d. The cost for teacher breaks is also made as a piece-wise linear function, using the new variable $vt_{t,d,j} \in \{0,1\}$ The following constraints are imposed to denote the number of undesirable breaks for a given teacher,

$$\sum_{g,s} V_{b,d} \cdot P_{g,t} \cdot x_{s,g,b} = p_{t,b,d} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{11.12}$$

$$|B|_d - (|B|_d - \mathrm{ord}(b)) \cdot p_{t,b,d} \geq h^{\mathrm{first}}_{t,d} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{11.13}$$

$$\mathrm{ord}(b) \cdot p_{t,b,d} \leq h^{\mathrm{last}}_{t,d} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{11.14}$$

$$h^{\mathrm{last}}_{t,d} - h^{\mathrm{first}}_{t,d} + 1 - \sum_{b} p_{t,b,d} = w_{t,d} \qquad \forall\, t, d \tag{11.15}$$

$$\sum_{j} vt_{t,d,j} \cdot \mathrm{ord}(j) = w_{t,d} \qquad \forall\, t, d \tag{11.16}$$

$$\sum_{j} vt_{t,d,j} = 1 \qquad \forall\, t, d \tag{11.17}$$

The contribution to the objective is as follows

$$-\sum_{s,d,j} \gamma_{s,j} \cdot vs_{s,d,j} - \sum_{t,d,j} \beta_{t,j} \cdot vt_{t,d,j} \tag{11.18}$$

11.3.3 Sequences

In connection with the undesirable breaks, the CTP also contains some needed breaks. For the students it is often necessary to give them a break between each consultation meeting due to traveling time between meeting rooms. This impose the following constraints.

$$\sum_{g} \left(x_{s,g,b} + x_{g,b_{+1},s}\right) \leq 1 \;\forall\, s, d, b \in B \backslash \{b_{|B_J|}\}, HS = 1, V_{b,d} = V_{b_{+1},d} = 1 \tag{11.19}$$

The teachers seldom change location during the consultation period, so travel time is not needed. However, as mentioned it is undesirable for the teachers to have a long sequence of meetings, as they need a break now and then. The maximum size of a sequence for a teacher is denoted $Q \in \mathbb{N}$. Let the variable $y_{t,b,d} \in \{0,1\}$ take value 1 if time slot b is the start of a sequence of length greater than Q on day d for teacher t. The following equation constraints this variable,

$$\sum_{s} \sum_{\substack{b'=b \\ V_{b',d}=1}}^{b+Q} p_{t,b',d} - y_{t,b,d} \leq Q \qquad \forall\, t, d, b \in B \backslash \{b_j | j > |B| - Q\}, V_{b,d} = 1 \tag{11.20}$$

The contribution to the objective is given by

$$-\sum_{t,b,d} \omega \cdot y_{t,b,d} \tag{11.21}$$

11.3.4 Day distribution

In case the consultation has more than one day it is preferred that each student and each teacher only has meetings on a single day. $u_{t,d} \in \{0,1\}$ and $u_{s,d} \in \{0,1\}$ denotes if teacher t or student s has a meeting on day d, respectively. $v_t \in \mathbb{N}$ and $v_s \in \mathbb{N}$ denotes the number of days where teacher t and student s have meetings, respectively. The number of days with meetings is punished in the objective by

$$-\sum_{s} \zeta_s \cdot v_s - \sum_{t} \zeta_t \cdot v_t \tag{11.22}$$

and is constrained by the following

$$\sum_{g} V_{b,d} \cdot x_{s,g,b} \leq u_{s,d} \qquad \forall\, s, b, d \tag{11.23}$$

$$\sum_{s,g} V_{b,d} \cdot P_{t,g} \cdot x_{s,g,b} \leq u_{t,d} \qquad \forall\, t, b, d \tag{11.24}$$

$$\sum_{d} u_{s,d} - 1 \leq v_s \qquad \forall\, s \tag{11.25}$$

$$\sum_{d} u_{t,d} - 1 \leq v_t \qquad \forall\, t \tag{11.26}$$

11.3.5 IP model for CTP

The entire model for CTP is given in (11.27).

Consultation Timetabling Problem IP (11.27)

$$\max \sum_{s,g,b}\left(\alpha_{s,g,b}-\sum_{t}\delta_t\cdot D_{t,b}\cdot P_{g,t}+\delta_s\cdot E_{s,b}\right)\cdot x_{s,g,b}-\sum_{s,d,j}\gamma_{s,j}\cdot vs_{s,d,j}-\sum_{t,d,j}\beta_{t,j}\cdot vt_{t,d,j} \quad (11.27a)$$
$$-\sum_{t,b,d}\omega\cdot y_{t,b,d}-\sum_{s}\zeta_s\cdot v_s-\sum_{t}\zeta_t\cdot v_t$$

s.t.

$$\sum_{b} x_{s,g,b} \le R_{s,g} \quad \forall\, s,g \quad (11.27b)$$
$$\sum_{g} x_{s,g,b} \le C_{s,b} \quad \forall\, s,b \quad (11.27c)$$
$$\sum_{s,g} P_{g,t}\cdot x_{s,g,b} \le 1 \quad \forall\, t,b \quad (11.27d)$$
$$\sum_{g}\left(x_{s,g,b}+x_{g,b_{+1},s}\right) \le 1 \quad \forall\, s,d,b\in B\backslash\{b_{|B_J|}\}, HS=1, V_{b,d}=V_{b_{+1},d}=1 \quad (11.27e)$$
$$\sum_{s}\sum_{\substack{b'=b\\ V_{b',d}=1}}^{b+Q} p_{t,b',d}-y_{t,b,d} \le Q \quad \forall\, t,d,b\in B\backslash\{b_j|j>|B|-Q\}, V_{b,d}=1 \quad (11.27f)$$
$$\sum_{g} V_{b,d}\cdot x_{s,g,b} = n_{s,b,d} \quad \forall\, s,b,d,V_{b,d}=1 \quad (11.27g)$$
$$|B|_d-(|B|_d-\mathrm{ord}(b))\cdot n_{s,b,d} \ge f^{\mathrm{first}}_{s,d} \quad \forall\, s,b,d,V_{b,d}=1 \quad (11.27h)$$
$$\mathrm{ord}(b)\cdot n_{s,b,d} \le f^{\mathrm{last}}_{s,d} \quad \forall\, s,b,d,V_{b,d}=1 \quad (11.27i)$$
$$f^{\mathrm{last}}_{s,d}-f^{\mathrm{first}}_{s,d}+1-\sum_{b} n_{s,b,d}\cdot(1+HS)+HS = z_{s,d} \quad \forall\, s,d \quad (11.27j)$$
$$\sum_{j} vs_{s,d,j}\cdot \mathrm{ord}(j) = z_{s,d} \quad \forall\, s,d \quad (11.27k)$$
$$\sum_{j} vs_{s,d,j} = 1 \quad \forall\, s,d \quad (11.27l)$$
$$\sum_{g,s} V_{b,d}\cdot P_{g,t}\cdot x_{s,g,b} = p_{t,b,d} \quad \forall\, t,b,d,V_{b,d}=1 \quad (11.27m)$$
$$|B|_d-(|B|_d-\mathrm{ord}(b))\cdot p_{t,b,d} \ge h^{\mathrm{first}}_{t,d} \quad \forall\, t,b,d,V_{b,d}=1 \quad (11.27n)$$
$$\mathrm{ord}(b)\cdot p_{t,b,d} \le h^{\mathrm{last}}_{t,d} \quad \forall\, t,b,d,V_{b,d}=1 \quad (11.27o)$$
$$h^{\mathrm{last}}_{t,d}-h^{\mathrm{first}}_{t,d}+1-\sum_{b} p_{t,b,d} = w_{t,d} \quad \forall\, t,d \quad (11.27p)$$
$$\sum_{j} vt_{t,d,j}\cdot \mathrm{ord}(j) = w_{t,d} \quad \forall\, t,d \quad (11.27q)$$
$$\sum_{j} vt_{t,d,j} = 1 \quad \forall\, t,d \quad (11.27r)$$
$$\sum_{g} V_{b,d}\cdot x_{s,g,b} \le u_{s,d} \quad \forall\, s,b,d \quad (11.27s)$$
$$\sum_{s,g} V_{b,d}\cdot P_{t,g}\cdot x_{s,g,b} \le u_{t,d} \quad \forall\, t,b,d \quad (11.27t)$$
$$\sum_{d} u_{s,d}-1 \le v_s \quad \forall\, s \quad (11.27u)$$
$$\sum_{d} u_{t,d}-1 \le v_t \quad \forall\, t \quad (11.27v)$$
$$x_{s,g,b}\in\{0,1\},\ y_{t,b,i}\in\{0,1\} \quad (11.27w)$$
$$w_{t,d}\in\mathbb{N},\ z_{s,d}\in\mathbb{N} \quad (11.27x)$$
$$vt_{t,d,j}\in\{0,1\},\ vs_{s,d,j}\in\{0,1\} \quad (11.27y)$$
$$f^{\mathrm{first}}_{s,d}\in\mathbb{N},\ f^{\mathrm{last}}_{s,d}\in\mathbb{N},\ h^{\mathrm{first}}_{t,d}\in\mathbb{N},\ h^{\mathrm{last}}_{t,d}\in\mathbb{N} \quad (11.27z)$$
$$p_{t,b,d}\in\{0,1\},\ n_{s,b,d}\in\{0,1\} \quad (11.27aa)$$
$$u_{s,d}\in\{0,1\},\ u_{t,d}\in\{0,1\} \quad (11.27ab)$$
$$v_s\in\mathbb{N},\ v_t\in\mathbb{N} \quad (11.27ac)$$

11.3.6 Complexity

In the following a proof of $\mathcal{NP}$-hardness is given by showing that a well known NP-hard problem, the Graph Coloring Problem (GCP), is polynomially reducible to CTP.

An arbitrary instance of GCP consists of a graph $G = (V, E)$ and a number of colors k. The decision-version of the GCP asks the following: Does graph G admit a proper vertex coloring with k colors, such that no adjacent vertices take the same color?

To answer this question we solve a CTP with parameters $\delta_t = \delta_s = \gamma_{s,j} = \beta_{t,j} = \omega = \zeta_s = \zeta_t = HS = 0, \alpha_{s,g,b} = 1, C_{s,b} = 1$. This makes all constraints redundant, except for (11.27b), (11.27c) and (11.27d). Further assuming every student has exactly one request, $\sum_g R_{s,g} = 1 \, \forall s$, makes constraint (11.27c) redundant.

For each vertex $v \in V$ in graph G, create a student s and a teachergroup g, and let the meeting request (s, g) represent vertex v. The set of vertices is hence represented by a setting of parameter $R_{s,g}$. If vertex $v_1 = (s_1, g_1)$ and vertex $v_2 = (s_2, g_2)$ are adjacent in graph G, create a teacher t and assign it to both g_1 and g_2, i.e. $P_{g_1,t} = P_{g_2,t} = 1$. I.e. every teacher will have exactly two meeting requests. Let the set of time slots B represent the set of colors (such that $|B| = k$).

Hence the GCP-instance is represented by the following CTP instance:

$$\max \quad \sum_{s,g,b} x_{s,g,b} \tag{11.28a}$$

$$\text{s.t.} \quad \sum_{b} x_{s,g,b} \leq R_{s,g} \quad \forall \, s, g \tag{11.28b}$$

$$\sum_{s,g} P_{g,t} \cdot x_{s,g,b} \leq 1 \quad \forall \, t, b \tag{11.28c}$$

$$x_{s,g,b} \in \{0, 1\} \tag{11.28d}$$

Constraint (11.28b) specifies that each vertex (meeting request) can at most be assigned one color (time slot). Constraint (11.28c) specifies that no teacher can be assigned more than one meeting in each time slot, which specifies that no adjacent vertices can take the same color.

To answer the question whether G is k-colorable, solve the CTP instance and check if all meeting requests are assigned a time slot, i.e. $\sum_b x_{s,g,b} = 1 \, \forall s, g, R_{s,g} = 1$. If so the answer is yes, otherwise the answer is no. Hence the Graph Coloring Problem is polynomially reducible to CTP, and CTP is therefore $\mathcal{NP}$-hard.

11.3.7 Defining weights

In the following the weights of the model are selected due to the preferences of the Danish high schools. MaCom A/S has greatly assisted this process. Table 11.1 lists all the weights in the model and their priority.
From analysis of previous consultations in the Danish high schools, it is noticed that the students rarely request more than five teacher groups for consultations. And even though the students have the opportunity to request more than five, they seldom use this option. From this analysis it is chosen to stick the request with priority higher than five to the same weight. This gives the

Table 11.1: Weight prioritizing

Weight	Symbol	Priority	Value dependency
(Max) Request fulfilling	$\alpha_{s,g,b}$	Very High	Priority of request (s,g) in time slot b
(Min) Teacher holes	$\beta_{t,j}$	High	Amount of requests of teacher t
(Min) Teacher sequence violation	ω	High	N/A
(Min) Teacher activities interruption	δ_t	Medium	N/A
(Min) Teacher multiple days	ζ_t	Medium	N/A
(Min) Student holes	$\gamma_{s,j}$	High	Amount of requests of student s
(Min) Student activities interruption	δ_s	Medium	N/A
(Min) Student multiple days	ζ_s	Medium	N/A

following function for the request weights $\alpha_{s,g,b}$:

$$\alpha_{s,g,b} = \begin{cases} \kappa_b + 12 - 2 \cdot (i-1) & i \leq 5 \\ \kappa_b + 2 & i \geq 6 \end{cases} \tag{11.29}$$

where $i \in \mathbb{Z}^+$ is the priority of request (g,s). Furthermore there is a set-point for each day given by b_d^* for which it is desired that the schedule plan for the given day is centered around. Let κ_b denote the penalty for assigning a request to time slot b, defined by

$$\kappa_b = -\frac{\sum_d V_{b,d} \cdot |b_d^* - b|}{|B|} \tag{11.30}$$

The cost of an undesirable break for a teacher, $\beta_{t,j}$, is defined as follows,

$$\beta_{t,j} = \begin{cases} 0 & j = 0 \\ j & j \geq 1 \wedge SCTP \\ j^{1+\frac{1.5}{\eta_t}} & j \geq 1 \wedge PCTP \end{cases} \tag{11.31}$$

where η_t is the number of requests for teacher groups where teacher t is a group member, $\eta_t = \sum_{s,g} P_{g,t} \cdot R_{s,g}$. I.e. $\beta_{t,j}$ depends on the number of requests for the given teacher t. The distribution of $\beta_{t,j}$ is chosen such that a teacher which few students have requested is given a high penalty for undesirable breaks. Likewise, a teacher with many requests has a low penalty for undesirable breaks. This is due to the fact that teachers with many requests will most likely have a more dense schedule, and are therefore not too picky about additional breaks. The reason why there is a difference between the weights for the different consultations types is due to the consultation interval. For the PCTP the consultation meetings are normally located in the evening, and hence we want to penalize the undesirable breaks. The SCTP is typically taken place in the daytime, i.e. the teachers are already at the high school, hence undesirable breaks are not that significant. The weight of an undesirable break for a student $\gamma_{s,j}$ is analogues,

$$\gamma_{s,j} = \begin{cases} 0 & j = 0 \\ j^{1+\frac{2}{\eta_s}} & j \geq 1 \end{cases} \tag{11.32}$$

where η_s is the number of requests of student s.
The cost for violating the length of a sequence for a teacher is given by ω.

$$\omega = 2 \tag{11.33}$$

In our model of the CTP the high school administration selects if interrupting other activities of the students or teachers is allowed. If this is not the case, the costs δ_t and δ_s are selected as infinitely high (implementation-wise the corresponding constraints are treated as hard-constraints). If interrupting other activities are not allowed, these costs are selected as a constant value,

$$\delta_t = \delta_s = \begin{cases} \infty & \text{Interrupting activities not allowed} \\ 4 & \text{Interrupting activities allowed} \wedge PCTP \\ 1 & \text{Interrupting activities allowed} \wedge SCTP \end{cases} \tag{11.34}$$

Like for the undesirable break cost $\beta_{t,j}$, we distinguish between the two consultations types. For the PCTP it is expensive to interrupt other activities since it is held in the evening and hence other activities are typically other types of meetings. The SCTP is held in the daytime, and it is allowed to 'lent' a student from a lecture for a small cost.

11.4 Adaptive Large Neighborhood Search

In this section a heuristic alternative to solving the IP-model (11.27) is described. The performance of these two methods are compared in Section 11.6.

As the local search algorithm we have chosen to use the *Large Neighborhood Search* (LNS) proposed by Shaw (1998). Most local search algorithms explicitly defines the neighborhood, but the neighborhood in LNS is defined implicitly by a *destroy* and a *repair* method. The neighborhood of a solution is then defined as the set of solutions that can be reached by first applying a destroy method and then a repair method. In this article we will use Adaptive Large Neighborhood Search (ALNS), in which the LNS is extended by multiple destroy and repair methods. ALNS was first described in Pisinger and Ropke (2005), and has since been used with success on various problems, especially variants of Vehicle Routing Problems (VRP), see e.g. Ropke and Pisinger (2006); Laporte et al. (2010); Azi et al. (2010); Ribeiro and Laporte (2012); Lei et al. (2011). A pseudo-code for the ALNS heuristic is shown in Algorithm 1.

Algorithm 1 Adaptive Large Neighborhood Search

1: **input:** a feasible solution $x^s_{g,b}$
2: solution $x_{best} = x$, $\pi = (1, \ldots, 1)$
3: **while** stop-criterion not met **do**
4: $x' = x$
5: select destroy and repair methods $d \in \Omega^-$ and $r \in \Omega^+$ using π
6: select $q \in \mathbb{N}$
7: remove q requests from x' using d
8: reinsert removed requests into x' using r
9: **if** $c(x') > c(x_{best})$ **then**
10: $x_{best} = x'$
11: **end if**
12: **if** accept(x', x) **then**
13: $x = x'$
14: **end if**
15: update π
16: **end while**
17: return x_{best}

The sets of destroy and repair methods are denoted Ω^- and Ω^+, respectively. The variable π, which stores the weight of all destroy and repair methods, is introduced in line 2. Initially all methods have the same weight. In line 5 the weight parameter π is used to select the destroy and repair methods. In line 12 an accept function evaluates if the new solution should become the new current solution. The accept function can be implemented in different ways. We have chosen to implement a Simulated Annealing-like acceptance criterion, which will be described later.

An ALNS framework has the advantage of using different neighborhoods, such that the algorithm hopefully explores a large part of the solution space. For more information regarding ALNS we recommend Ropke and Pisinger (2006) and Pisinger and Ropke (2010).

11.4.1 ALNS scoring scheme

A central part of the ALNS algorithm is the scoring scheme of destroy and repair methods. A scoring scheme can essentially be characterized by two central topics; 1) How to quantify the performance of each heuristic. 2) The reaction factor, i.e. how sensitive is the selection process to recent records of performance.

We adapt a scoring scheme based on the technical report of Muller and Spoorendonk (2010), where performance is tracked by the percentage-wise gap between the new found solution and the current solution. This scoring scheme has the advantage of having few parameters to tune, and using the gap between solutions seems as a intuitively good way of measuring heuristic performance. Below the scoring scheme is explained in details.

Runs of the algorithm is divided into segments $\{t_0, t_1, \ldots, t_n\}$ each consisting of N_{it} iterations. Let π_i^t be the weight of heuristic i in segment t. The probability of choosing heuristic i in segment t is $\frac{\pi_i^t}{\sum_j \pi_j^t}$. At the end of each segment t, the following update is performed for all heuristics,

$$\pi_i^{t+1} = \rho \frac{\bar{\pi}_i^t}{a_i^t} + (1-\rho)\pi_i^t \tag{11.35}$$

where a_i^t is the number of times heuristic i has been selected in segment t. $\bar{\pi}_i^t$ is the *observed* weight of heuristic i in segment t, which in each iteration is incremented depending on the quality of the new found solution. $\rho \in [0,1]$ is the reaction factor. A high reaction factor means that the weights of a segment will be very dependent upon the observed weights of the previous segment.

The observed weight $\bar{\pi}_i^t$ is updated in each iteration. Let x be the current solution, and x' the new found solution by applying neighborhood i. In the technical report Muller and Spoorendonk (2010) the following formula is used,

$$\text{gap} = \frac{c(x') - c(x)}{c(x)} \tag{11.36}$$

$$\bar{\pi}_i^t = \bar{\pi}_i^t + m^{\text{gap}} \tag{11.37}$$

where m is a constant. We will use a slightly changed version of this formula, since we have observed that the gap formulated by (11.36) most often yields values of magnitude $\pm 10^{-4}$, meaning that the observed weight $\bar{\pi}_i^t$ will rarely change value of significant magnitude. Therefore we introduce a scale parameter in the formula,

$$\bar{\pi}_i^t = \bar{\pi}_i^t + m^{\min(\sigma \cdot \text{gap}, 1)} \tag{11.38}$$

where $\sigma \in \mathbb{R}^+$ is a parameter that needs tuning. We fix $m = 5$ and rely on the parameter tuning to set a suitable value for σ. The min-operator in the exponent of m is necessary to ensure the weight stay within a reasonable interval, in case we hit an iteration where the scaled gap is big and positive.

11.4.2 Request removal

The ALNS heuristic for the CTP makes use of two different removal heuristics, each searching a given removal neighborhood. The heuristics takes as input a given solution $x_{s,g,b}$ and an integer $q \in \mathbb{N}$. The output of the heuristics is the solution where q meetings have been removed. The value of q is selected as a random number which satisfies,

$$3 \leq q \leq \max\left(\xi \cdot \sum_{g,s} R_{s,g}, 5\right) \tag{11.39}$$

where $\xi \in]0,1]$ is the maximum percentage of requests to remove. In accordance with Muller (2009) we decay ξ over time, starting with a high value ξ_{start} and ending with a smaller ξ_{end}. Given the runtime of the algorithm, we divide it into 100 segments such that ξ is decreased by $\frac{\xi_{\text{start}} - \xi_{\text{end}}}{100}$ in each segment. This decay of ξ means that the size of the searched neighborhood is progressively reduced. This has the advantage of only performing small changes towards the end of the solution process, where we expect a good solution has been found.

11.4.2.1 Random removal

The simplest removal heuristic, which randomly removes q meetings from the solution. This simple heuristic obviously has the effect of diversifying the search.

11.4.2.2 Shaw removal

This removal heuristic was introduced in Shaw (1997, 1998) where it is used on the VRP. In this section the heuristic is modified to suit the CTP. The general idea of the heuristic is to remove meetings which are somehow related, since there is a good chance that such requests can swap positions and possibly improve the solution. In this paper two factors determine if meeting i is related to meeting j: Similarity between students, and similarity between teachers. S_i and T_i indicates the set of students and teachers of meeting i, respectively. Notice that a meeting always contains exactly one student, i.e. $|S_i| = 1$. Let the measure of relatedness between meeting i and j be defined by $\mathcal{M}(i,j) \in [0,1]$,

$$\mathcal{M}(i,j) = \frac{|T_i \cap T_j| + |S_i \cap S_j|}{\min(|T_i|, |T_j|) + 1} \tag{11.40}$$

I.e. relatedness is the percentage of individuals which is shared between the meetings, such that a high value of $\mathcal{M}$ means that the meetings are very related. This simple formulation of relatedness is done without any additional parameters. An alternative natural formulation would be to scale the student-relatedness and teacher-relatedness by two independent parameters. However we have chosen the shown formula due to its simplicity. An addition to the formula could be to introduce a term which determines time slot relatedness, although it should be noted that relatedness between slots is only directly relevant if there also exists some relatedness between students or teachers.

A pseudo code for Shaw removal is shown in Algorithm 2.

To avoid the situation where the same meetings are removed over and over, the algorithm is randomized. The level of randomness is controlled by the parameter $p_{\text{shaw}} \in \mathbb{R}^+, p_{\text{shaw}} \geq 1$. This means that p_{shaw} somehow defines how random the element is chosen, where $p_{\text{shaw}} = 1$ corresponds to completely random.

Algorithm 2 Shaw removal

input: A feasible solution $x_{s,g,b}$**,** $q \in \mathbb{N}$**,** $p_{\mathbf{shaw}} \in \mathbb{R}^+$
request: r = a randomly selected meeting from $x_{s,g,b}$
set of requests : $D = \{r\}$
while $|D| < q$ **do**
    r = randomly selected meeting from D
    L = all meetings from $x_{s,g,b}$ not in D, sorted by decreasing similarity to r
    choose a random number $y^{p_{\text{shaw}}} \in [0, 1[$
    l = element number $y^{p_{\text{shaw}}} \cdot |L|$
    $D = D \cup L[l]$
end while
remove the meetings in D from $x_{s,g,b}$

11.4.3 Repair heuristic

The repair heuristics are given a set of consultation meetings and a set of not granted meeting-requests.

11.4.3.1 Basic greedy heuristic

A trivial algorithm for the CTP is a simple greedy algorithm which places one request at a time in order of contribution to the objective. In each iteration of the algorithm this process is repeated until no more requests which improves the solution can be inserted. Implementation wise the algorithm suffers from cost-dependencies, since the contribution of inserting each request possibly changes after each insertion. This is slightly optimized by only recalculating the cost of those requests which the last insertion can possibly effect. I.e. recalculate the cost of those requests which has the same student as the last insertion, or if the teacher group overlaps with the one of the last insertion. This repair heuristic is used to create an initial feasible solution for the CTP.

11.4.3.2 Regret heuristics

The regret heuristic improves the basic greedy by incorporating a kind of look-ahead information when selecting a request to insert. Informally speaking, the heuristic aims at inserting the request which we will regret most if not inserted immediately. The regret heuristic has been used by Potvin and Rousseau (1993) and Pisinger and Ropke (2005) for the *Vehicle Routing Problems with Time Windows*. Let c_r^k denote the change in the objective value by inserting request r into the k^{th} best position. E.g. c_r^2 denotes the change in the objective value by inserting request r in the second best position. A Regret-2 heuristic will in each iteration choose to insert the request r where the difference between best and second best position is largest, i.e.

$$r := \underset{r \in R_g^s, c_r^1 > 0}{\arg\max} \left(c_r^1 - c_r^2\right) \tag{11.41}$$

The request r is inserted at its best position, so we restrict the heuristic to only look at requests where the best position is actually feasible and yields a positive change in objective. This restriction is necessary since the objective of the CTP contains both a minimization and a maximization part, and we are not interested in inserting requests which have negative impact on the objective. The heuristic can be extended by looking at k positions for each request. The

request to insert is then chosen according to

$$r := \underset{r \in R_g^s, c_r^1 > 0}{\arg\max} \sum_{h=2}^{k} \left(c_r^1 - c_r^h\right) \tag{11.42}$$

We will in this paper incorporate the regret heuristic for several choices of k. The basic greedy algorithm from the previous section is a Regret-1 heuristic due to the tie-breaking rules. For a Regret-1 heuristic the most profitable request is inserted in each iteration. Most papers distinguish between Regret-1 and other regret heuristic, however implementation wise they are not very different. Setting $k = |B|$ corresponds to the full Regret-k heuristic.

Even though the regret heuristic is designed for VRP, it seems well suited for the CTP due to its assignment character. It seems valuable to attempt to predict which request is the most critical to insert. By some basic tests, we have chosen to use Regret-2, Regret-3, and Regret-$|B|$ as insertion heuristics.

11.4.4 Algorithm setup

According to Ropke and Pisinger (2006), using myopic repair heuristics, like those of this paper, one may apply noise to the objective function to obtain a more efficient algorithm. By applying noise, the repair heuristic will not always make the move that seems best locally. Ropke and Pisinger (2006) support this by strong computational results. However, preliminary tests show that, in our case, adding noise does not yield a more efficient algorithm. More precisely, noise was added such that it was controlled by a linear-scale parameter, and excessive tuning on this parameter yielded no convergence at all. I.e. this parameter had (close to) no impact on the algorithm efficiency. A similar result for the *Cumulative Capacitated Vehicle Routing Problem* is reported in Ribeiro and Laporte (2012).

In occurrence with Ropke and Pisinger (2006) we borrow an acceptance criteria from Simulated Annealing. A solution x is always accepted if $c(x) > c(x_{\text{best}})$. If $c(x) < c(x_{\text{best}})$ then x is accepted with probability
$\exp\left(-\frac{c(x_{\text{best}})-c(x)}{T}\right)$. In each iteration the temperature T is updated by $T = d_{\text{SA}}T$, where $0 < d_{\text{SA}} < 1$. Giving the *temperature control parameter* w_{SA}, $0 < w_{\text{SA}} < 1$, T is initially selected such that a solution is accepted with probability $\frac{1}{2}$ if its change in objective is w_{SA} percent worse than the initial solution x_0, i.e.

$$\exp\left(-\frac{(c(x_0) - (1 - w_{\text{SA}}) \cdot c(x_0))}{T_0}\right) = \tfrac{1}{2} \quad \Rightarrow \quad T_0 = \frac{w_{\text{SA}} \cdot c(x_0)}{\ln(2)} \tag{11.43}$$

This has the advantage of better adapting the temperature to each dataset.

Furthermore at the start of each segment (those of the ALNS scoring scheme), the current solution is set to the current best.

11.5 Parameter tuning

The proposed heuristic contains many free parameters. It is essential that these parameters are tuned to achieve good performance, see e.g. Adenso-Diaz and Laguna (2006) and Diao et al. (2003). Tuning of metaheuristics is usually done by rules-of-thumb and the researchers personal experience. However some well performing automated algorithms have lately been introduced, mainly *ParamILS* (Hutter et al. (2009)) and *Race*-algorithms (Birattari (2005)). In this paper we will use the *F-Race* algorithm for tuning, as its implementation burden seems light, and it

has proven competitive for some heuristic methods, see Montero et al. (2010) and Pellegrini et al. (2010).

The main idea of a race algorithm is to sequentially process a set of data instances using all possible parameter configurations. In each iteration, the parameter configurations which are statistically inferior are eliminated. The algorithm is ran until one parameter configuration remains or the specified time limit is exceeded, see Algorithm 3. If more than one parameter configuration remains once the algorithm terminates, the one which in average has performed best is selected. The advantage of a race algorithm is that bad parameter configurations are eliminated early, such that no more valuable computation time is spend on evaluating these. The racing algorithm differs from most other tuning approaches in the sense that it only performs one algorithm run per parameter configuration per data instances. This relies on the proof in Birattari (2005) where it is shown that this is the optimal experimental setting in terms of variance of the estimated performance.

Algorithm 3 Race Tuning

input:
  Θ: Set of parameter configurations
  T_{exp}: Computation time of each experiment
  T_{total}: Time limit
  α: Confidence level
$i = 0, S_\theta = \Theta, C_\theta = \emptyset$
while $t < T_{\text{total}}$ AND $|S_\theta| > 1$ **do**
  dataset = RandomSampled()
  for all $\theta \in S_\theta$ **do**
    C_θ^i = EvalSolution(T_{exp}, θ, dataset)
  end for
  $i = i + 1$
  Drop inferior parameter configurations from S_θ by statistical test, using confidence level α
end while

In a F-Race algorithm, the *Friedman Two-way Analysis of Variance by Ranks* test is used to determine whether there is sufficient statistically evidence to eliminate parameter configurations from future iterations. If this is the case, then post-tests are performed where pairwise comparison between the best candidate and the remaining determines which configurations should be eliminated, if any. The F-Race algorithm has been successfully used for tuning in a number of cases, see e.g. Chiarandini et al. (2006); Becker et al. (2006); Pellegrini and Birattari (2007).

A problem of the described Racing algorithm, which applies to most tuning frameworks, is the so-called full-factorial design, meaning that the full set of parameter configuration is initially considered. This results in the F-Race becoming impractical and computational prohibitive, if there exists a large number of parameters and each parameters can take a modest number of values. This has been addressed in Balaprakash et al. (2007) by defining a probabilistic model on the set of all possible parameter configurations, such that a small set of parameter configurations is generated in each iteration of the tuning process. Elite configurations are used to update the model to bias the search around high quality parameter configurations. This version of F-Race is denoted *Iterative F-Race* (I/F-Race).

In this paper we use a simplified I/F-Race algorithm, where we start out with a small subset of parameter configurations, and based on the Race-results of these we manually construct new configurations, which are believed to be superior. One could think of this approach as a sort of

manual iterative F-Race. Table 11.2 shows the best found parameter configuration. It should be mentioned that we set $b_{\text{SP}} = b_0$ and $\delta^s = \delta^t = \infty$, since these are the most common values chosen by the users. The datasets used are of the school year 2011/2012. w_{SA} is the temperature control

Table 11.2: Final values of tuned parameters, found by the F-Race algorithm with confidence level $\alpha = 0.05$.

Parameter	w_{SA}	d_{SA}	N_{it}	ρ	σ	ξ_{start}	ξ_{end}	p_{shaw}
Value	0.01	0.99	100	0.50	1000	0.30	0.0033	20

parameter and d_{SA} is the decay parameter for the SA based acceptance criteria. N_{it} defines the number of iterations between resets. ρ and σ are reaction factor and the scale parameter for the ALNS scoring scheme, respectively. ξ_{start} and ξ_{end} are the destroy percentage in the beginning and in the end of the running time. Finally, p_{shaw} indicates how random the element is chosen in the Shaw removal.

11.6 Performance

The goal of this section is to evaluate the performance of the developed solution methods, the ALNS algorithm and solving the IP model. Also a comparison with the existing heuristic of Lectio is made. All tests are performed using nUnit 2.6 in C# 4.0 on a machine with an Intel i7-930@2.8GHz CPU and 12GB of RAM. No parallelization has been implemented.

11.6.1 Performance comparison between ALNS and Gurobi

In the following, the performance of the state-of-the-art MIP solver Gurobi 5.01 (currently top-ranked in the MIP benchmark of Mittelman (2013)) and the implemented ALNS algorithm are compared. For both the PCTP and the SCTP, 100 datasets from the school-year 2011/2012 are selected from the database of Lectio.

In this experimental setup, the ALNS algorithm is run for 2 minutes. This low running time is due to the following: 1) The schools does generally not expect an algorithm to run longer, as they are usually not aware that it is a hard problem to solve. Some even believe the problem is trivial. 2) The ALNS "tailors-off" after a while, i.e only minor improvements are seen on the best found solution after the 2 minute mark.

The Gurobi solver is run for 1 hour, because we do not only want to evaluate the performance in terms of best found IP solution, we also want a good upper bound for the instances.

In the performance tests it is not allowed to interrupt other activities for the PCTP, i.e. $\delta_t = \delta_s = \infty$. For the SCTP interrupting activities is allowed. This is due to the fact that PCTP is normally arranged in the evening while SCTP is during the normal work-hours.

From Table 11.3 it is seen that ALNS in average finds solutions 4% from optimum. Even though ALNS has far lower running time than Gurobi, it finds better solutions in almost all cases.

Table 11.3: Gurobi 5.01 and ALNS for the PCTP on 100 datasets. For each dataset is listed the number of time slots "$|B|$", the total number of meeting requests "$\sum R$", the average number of requests pr. student, and the average number of requests pr. teacher. For Gurobi is shown the final objective value "x", the best bound found "UB", and the reported gap between these two. For ALNS, column "$\bar{x}$" is the mean performance of the algorithm over 10 runs, and column "σ" is the standard deviation for these runs. Finally, column "Gap" is the gap of mean performance and the upper bound found by Gurobi. The best found solution is marked with bold for each instance.

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Alleroed	12	51	3.0	2.4	**485.0**	485.0	0.0	484.8	0.1	0.0
Alssund	18	84	2.5	4.0	**850.6**	850.6	0.0	849.4	0.6	0.1
Aurehoej1	18	537	4.0	9.9	3270.3	3774.3	15.4	**3655.9**	6.9	3.2
Aurehoej2	18	409	3.8	6.6	3033.1	3300.5	8.8	**3219.5**	3.9	2.5
Broenderslev	24	241	4.0	4.8	1656.2	1965.1	18.7	**1905.6**	3.0	3.1
CPHWEST	39	133	3.7	5.8	990.5	1124.7	13.5	**1044.5**	3.0	7.7
DetKristne	32	247	4.1	11.2	1553.3	1978.5	27.4	**1830.9**	4.9	8.1
Dronninglund1	30	108	4.9	7.7	726.0	801.5	10.4	**782.0**	4.1	2.5
Dronninglund2	30	94	4.1	8.6	636.6	672.7	5.7	**664.5**	0.6	1.2
Egedal	27	408	3.4	6.3	3143.5	3625.0	15.3	**3558.2**	3.7	1.9
Egaa	24	265	3.2	9.8	2101.7	2374.0	13.0	**2318.5**	7.7	2.4
Esbjerg1	24	345	3.9	9.6	2306.2	2465.3	6.9	**2402.1**	4.2	2.6
Esbjerg2	24	307	3.5	4.3	2117.6	2439.9	15.2	**2314.3**	2.4	5.4
Esbjerg3	24	255	3.8	9.4	1770.0	1888.1	6.7	**1839.5**	2.8	2.6
Esbjerg4	24	351	4.0	9.2	2493.8	2700.5	8.3	**2612.3**	4.5	3.4
Frederikssund	24	49	3.3	4.5	**404.2**	406.8	0.7	403.8	0.4	0.8
Frederiksvaerk	8	74	2.4	3.2	**699.0**	699.0	0.0	697.9	0.6	0.2
Gefion	18	479	3.1	7.2	3316.6	4248.9	28.1	**3958.1**	23.7	7.4
Gladsaxe	40	901	4.1	11.0	5516.0	7163.7	29.9	**6950.7**	15.1	3.1
Greve	18	336	4.5	4.4	2133.1	2535.9	18.9	**2482.6**	4.4	2.2
Haslev1	18	123	2.9	5.4	1051.5	1077.1	2.4	**1060.2**	0.5	1.6
Haslev2	18	122	3.2	4.2	983.5	1019.6	3.7	**988.7**	2.6	3.1
Herlufsholm1	24	143	4.3	14.3	853.1	918.9	7.7	**894.5**	1.2	2.7
Herlufsholm2	24	88	4.9	6.8	599.5	671.3	12.0	**621.8**	1.9	8.0
Herning1	27	118	3.8	3.6	0.0	0.0	0.0	0.0	0.0	0.0
Herning2	27	75	3.4	6.3	0.0	0.0	0.0	0.0	0.0	0.0
Herning3	27	140	3.7	4.5	2.0	2.0	0.0	2.0	0.0	0.0
Himmelev	34	453	3.7	8.4	3118.1	3736.1	19.8	**3471.7**	8.8	7.6
Hjoerring	30	179	3.8	2.1	751.2	1229.9	63.7	**1009.2**	15.3	21.9
HorsensGym	18	123	3.2	3.4	**1131.5**	1133.8	0.2	1129.4	1.1	0.4
HorsensStats	21	143	2.9	3.0	1217.8	1289.3	5.9	**1253.8**	1.9	2.8
Ikast-Brande	30	52	3.5	5.8	419.4	463.7	10.6	**449.7**	0.4	3.1
Johannesskolen1	24	165	4.5	7.9	1131.2	1296.8	14.6	**1188.3**	4.3	9.1
Johannesskolen2	24	97	3.7	7.5	701.4	786.3	12.1	**743.9**	5.0	5.7
Johannesskolen3	28	135	3.6	7.5	519.2	581.4	12.0	**519.9**	3.2	11.8
Kalundborg	27	299	3.0	4.8	2281.0	2587.0	13.4	**2458.3**	5.3	5.2

Continued on next page

Table 11.3 – continued from previous page

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Kolding	18	80	3.5	4.0	**721.4**	725.2	0.5	721.0	0.2	0.6
Langkær1	18	52	3.5	2.6	**471.1**	471.1	0.0	470.6	0.3	0.1
Langkær2	18	90	3.5	3.8	788.9	814.6	3.3	**805.9**	0.5	1.1
Middelfart	27	223	3.1	6.2	1682.5	1923.6	14.3	**1788.1**	7.4	7.6
Morsoe1	27	105	3.9	10.5	768.8	834.2	8.5	**804.2**	3.2	3.7
Morsoe2	27	113	4.7	10.3	732.9	815.2	11.2	**778.4**	4.2	4.7
Munkensdam1	21	256	3.6	4.7	1871.9	2251.6	20.3	**2198.6**	5.3	2.4
Munkensdam2	21	345	3.4	6.3	2352.8	2931.6	24.6	**2846.8**	6.7	3.0
NielsSteensens1	36	117	5.9	7.8	715.8	781.9	9.2	**757.2**	3.9	3.3
NielsSteensens2	30	328	7.5	17.3	1249.9	1764.9	41.2	**1656.6**	5.9	6.5
NielsSteensens3	30	365	7.9	20.3	1125.7	1895.7	68.4	**1800.3**	7.4	5.3
NielsSteensens4	30	234	6.7	13.8	1101.6	1230.0	11.7	**1144.4**	5.2	7.5
NielsSteensens5	30	263	6.3	17.5	1461.7	1634.2	11.8	**1557.1**	5.7	5.0
Noerre	18	422	2.7	7.8	3574.1	4031.6	12.8	**3944.5**	5.2	2.2
Nordfyns	23	192	2.6	6.6	1761.9	1863.5	5.8	**1795.5**	6.2	3.8
Nordsjaellands1	34	1187	6.4	25.3	6001.7	7018.5	16.9	**6597.3**	27.9	6.4
Nordsjaellands2	34	1038	6.6	23.1	2298.7	2626.1	14.3	**2453.2**	14.7	7.1
Nordsjaellands3	34	457	6.3	13.9	2225.4	2858.3	28.4	**2634.7**	7.8	8.5
Nordsjaellands4	34	163	4.9	9.6	1100.9	1210.0	9.9	**1172.4**	2.9	3.2
Nordsjaellands5	40	712	5.6	19.8	2543.1	4796.2	88.6	**4460.5**	16.6	7.5
Nordsjaellands6	34	780	6.1	19.0	2500.4	4899.0	95.9	**4612.2**	14.2	6.2
Nordsjaellands7	34	880	6.1	19.1	2763.3	3047.9	10.3	**2894.5**	6.3	5.3
Nordsjaellands8	34	23	1.6	3.3	**242.1**	242.1	0.0	241.9	0.3	0.1
Nordsjaellands9	34	949	6.2	22.1	3202.9	5519.0	72.3	**5037.1**	31.8	9.6
Nordsjaellands10	34	31	1.9	4.4	**272.2**	272.9	0.2	269.1	3.3	1.4
Nyborg	24	119	3.2	5.7	55.4	55.4	0.0	55.4	0.0	0.0
Nykoebing	24	182	3.0	3.2	1447.1	1502.2	3.8	**1483.1**	1.2	1.3
NZahles1	25	324	4.3	9.5	2116.2	2456.3	16.1	**2365.7**	5.5	3.8
NZahles2	24	301	4.1	8.9	2035.3	2280.1	12.0	**2217.8**	6.5	2.8
Odder	18	95	4.0	7.3	740.0	773.5	4.5	**762.7**	1.5	1.4
Odsherreds	21	193	3.4	5.0	1533.4	1619.6	5.6	**1595.4**	2.2	1.5
Risskov1	15	65	3.1	4.6	**539.7**	539.7	0.0	536.7	2.5	0.6
Risskov2	15	149	3.5	5.7	**1263.2**	1273.1	0.8	1256.9	2.9	1.3
Risskov3	15	181	3.7	6.2	**1396.9**	1406.8	0.7	1389.7	4.8	1.2
Roedkilde	18	266	3.6	9.2	2161.1	2352.9	8.9	**2325.2**	5.2	1.2
Roedovre	51	779	3.6	10.3	1513.8	2032.4	34.3	**1661.7**	20.1	22.3
Rosborg1	24	218	3.5	9.9	1742.7	1876.3	7.7	**1827.9**	5.4	2.7
Rosborg2	28	268	3.7	11.7	1960.9	2297.2	17.2	**2223.3**	11.4	3.3
Rosborg3	28	487	1.9	9.6	4568.5	4939.6	8.1	**4750.0**	10.1	4.0
Rosborg4	26	235	3.6	8.4	1713.8	2033.2	18.6	**1960.4**	11.1	3.7
Roskilde	48	263	3.7	6.4	1716.4	2251.7	31.2	**2112.8**	6.4	6.6
Rybners	24	267	3.1	6.9	1923.3	2472.3	28.5	**2402.5**	3.8	2.9
SanktAnnae	21	320	4.1	7.0	2115.0	2498.7	18.1	**2381.6**	5.6	4.9
Skive	36	220	3.6	12.9	1611.0	1903.2	18.1	**1850.3**	2.9	2.9

Continued on next page

Table 11.3 – continued from previous page

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Slagelse	30	85	3.0	3.9	**805.3**	805.4	0.0	802.6	0.8	0.4
Solroed1	16	341	3.3	7.4	2418.2	2468.7	2.1	**2436.8**	6.1	1.3
Solroed2	16	415	3.4	7.2	3083.3	3317.9	7.6	**3263.0**	5.2	1.7
Soroe	24	369	3.7	8.2	2587.1	3111.2	20.3	**2947.3**	12.1	5.6
Soroe	33	335	4.2	5.6	1594.3	2649.3	66.2	**2255.0**	12.3	17.5
Stenhus	18	221	4.3	3.2	0.0	0.0	0.0	0.0	0.0	0.0
Stoevring	24	62	3.7	4.4	**521.4**	521.4	0.0	520.5	0.5	0.2
Struer1	30	237	3.3	4.4	1596.8	1801.2	12.8	**1656.9**	2.1	8.7
Struer2	30	333	3.3	8.8	2301.3	2790.8	21.3	**2534.0**	7.9	10.1
Svendborg1	18	96	2.1	2.9	**991.4**	991.4	0.0	991.1	0.2	0.0
Svendborg2	18	134	2.6	4.5	**1289.0**	1289.1	0.0	1288.0	0.3	0.1
Taarnby	36	791	4.6	11.0	4397.7	5918.0	34.6	**5609.2**	17.7	5.5
UCH	32	104	1.0	17.3	**949.8**	949.8	0.0	922.6	0.0	2.9
ViborgGym1	30	206	4.3	5.2	1367.7	1486.5	8.7	**1434.0**	2.6	3.7
ViborgGym2	30	149	4.4	5.3	1094.0	1146.6	4.8	**1133.4**	0.9	1.2
ViborgGym3	30	294	3.7	4.6	2153.6	2275.5	5.7	**2211.7**	1.8	2.9
ViborgHandel	30	324	4.2	18.0	2111.8	2615.9	23.9	**2526.7**	9.7	3.5
ViborgKatedral	40	337	4.8	11.2	1935.5	2516.5	30.0	**2313.4**	8.6	8.8
Vordingborg1	16	315	3.8	6.3	2222.0	2358.8	6.2	**2304.3**	2.8	2.4
Vordingborg2	16	239	3.3	5.6	1867.5	1950.0	4.4	**1924.6**	2.0	1.3
Average	26	279	3.9	8.3			14.8			4.0
Max	51	1187	7.9	25.3			95.9			22.3

Table 11.4 shows the performance for the SCTP.

Table 11.4: Gurobi and ALNS for the SCTP on 100 datasets. Columns are defined in analogous way to Table 11.3. The average number of requests pr. student is not shown, as it is 1.0 in all cases.

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Aabenraa	60	226	4.3	**2461.2**	2495.7	1.4	2387.5	3.3	4.5
Broendby1	21	69	3.8	677.3	707.3	4.4	**683.2**	2.8	3.5
Broendby2	14	69	4.3	**770.2**	779.5	1.2	768.6	1.0	1.4
Broendby3	24	62	3.4	609.0	632.8	3.9	**614.8**	1.3	2.9
Broenderslev1	69	115	3.5	**1335.4**	1340.9	0.4	1302.4	2.1	3.0
Broenderslev2	102	115	3.5	**1253.4**	1272.5	1.5	1236.0	1.6	3.0
Christianshavns	43	210	4.8	**2329.7**	2423.7	4.0	2223.7	4.4	9.0
Dronninglund1	100	134	4.2	1442.7	1481.3	2.7	**1453.5**	2.1	1.9
Dronninglund2	60	134	4.3	**1553.3**	1561.9	0.6	1537.5	2.8	1.6
Egaa	29	214	4.8	2257.5	2457.1	8.8	**2376.2**	4.0	3.4
Falkoner1	30	64	3.4	**671.6**	679.0	1.1	668.0	2.6	1.6
Falkoner2	37	206	4.2	2188.3	2345.6	7.2	**2266.1**	5.0	3.5

Continued on next page

Table 11.4 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Falkoner3	30	64	3.4	664.9	672.3	1.1	664.9	0.0	1.1
Grenaa1	28	122	3.8	1280.7	1380.3	7.8	**1325.4**	3.1	4.2
Grenaa2	24	122	3.8	1249.2	1330.7	6.5	**1290.3**	3.8	3.1
Greve1	28	157	3.3	1575.1	1762.2	11.9	**1693.3**	3.2	4.1
Greve2	62	259	4.1	2805.4	3035.8	8.2	**2913.5**	7.4	4.2
Greve3	20	51	3.2	**566.4**	569.9	0.6	566.3	0.0	0.6
Gribskov1	74	182	4.1	**1867.8**	1914.6	2.5	1787.6	3.2	7.1
Herlev1	24	71	2.8	729.9	751.5	3.0	**730.2**	0.9	2.9
Herlev2	29	78	2.8	682.1	791.7	16.1	**750.2**	1.2	5.5
Hoeng1	21	66	3.5	607.9	686.5	12.9	**621.5**	0.4	10.5
Hoeng2	23	98	3.9	1029.8	1071.8	4.1	**1038.5**	2.2	3.2
Hoeng3	22	45	2.7	392.7	481.7	22.7	**408.3**	1.2	18.0
Hoeng4	23	56	2.6	589.1	612.4	4.0	**589.2**	1.8	3.9
Koebenhavns1	16	143	3.9	1239.8	1278.3	3.1	**1242.0**	6.1	2.9
Koebenhavns2	16	100	3.7	**786.4**	786.5	0.0	785.6	0.8	0.1
Koebenhavns3	16	100	3.7	725.8	725.9	0.0	725.8	0.0	0.0
Koebenhavns4	25	146	3.8	1402.3	1486.1	6.0	**1424.6**	0.8	4.3
Koege1	30	255	8.5	1348.2	2475.2	83.6	**2333.2**	10.3	6.1
Koege2	36	261	6.2	**2381.3**	2493.7	4.7	2045.7	5.2	21.9
Koege3	74	258	8.6	**2775.1**	2903.2	4.6	2622.9	5.0	10.7
Kolding1	24	219	5.0	2283.5	2430.9	6.5	**2348.7**	4.8	3.5
Kolding2	45	174	3.8	**1934.9**	2005.3	3.6	1908.8	4.1	5.1
Langkaer1	62	215	5.4	2195.8	2472.6	12.6	**2239.4**	7.8	10.4
Langkaer2	60	216	5.4	**2351.0**	2481.0	5.5	2240.1	9.3	10.8
Langkaer3	60	216	5.4	**2359.1**	2472.8	4.8	2258.3	6.6	9.5
Langkaer4	30	57	3.8	546.4	596.1	9.1	**566.2**	3.6	5.3
Langkaer5	56	217	5.6	**2282.3**	2493.5	9.3	2253.3	3.4	10.7
Langkaer6	62	56	3.7	**631.5**	652.6	3.3	629.2	0.7	3.7
Mariagerfjord1	29	123	4.0	1227.0	1387.2	13.1	**1318.6**	2.0	5.2
Mariagerfjord2	29	123	4.0	1269.0	1402.6	10.5	**1345.3**	4.0	4.3
Marselisborg1	22	102	3.4	1021.3	1087.4	6.5	**1045.4**	1.9	4.0
Marselisborg2	17	106	3.3	**1036.2**	1046.9	1.0	1035.4	1.0	1.1
Marselisborg3	22	105	3.9	1049.3	1156.3	10.2	**1098.3**	4.3	5.3
Marselisborg4	17	96	3.2	**948.6**	955.2	0.7	947.2	0.9	0.8
Munkensdam	43	191	5.6	**2179.9**	2203.8	1.1	2067.9	4.6	6.6
Nordfyns1	22	173	5.1	1871.9	1974.3	5.5	**1926.6**	1.6	2.5
Nordfyns2	21	173	5.2	1846.9	1975.6	7.0	**1929.9**	2.6	2.4
Nordfyns3	22	173	5.1	1831.5	1973.7	7.8	**1908.1**	2.1	3.4
Nordfyns4	21	173	4.1	1438.0	1538.2	7.0	**1478.3**	3.7	4.1
Noerresundby	31	303	4.7	2959.5	3437.4	16.2	**3291.7**	2.6	4.4
NZahles1	13	69	3.3	615.7	636.9	3.5	**619.1**	0.9	2.9
NZahles2	13	62	3.9	**512.1**	524.3	2.4	509.5	1.3	2.9
Odsherreds	49	119	3.8	**1365.1**	1372.6	0.6	1289.4	3.4	6.5
Oeregaard1	20	219	5.6	2257.3	2296.6	1.7	**2258.1**	5.6	1.7

Continued on next page

Table 11.4 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Oeregaard2	20	213	5.0	1728.9	1778.5	2.9	**1743.9**	3.2	2.0
Oeregaard3	20	219	5.6	2327.5	2372.3	1.9	**2340.2**	3.3	1.4
Oeregaard4	20	219	5.6	2338.4	2373.5	1.5	**2339.9**	2.6	1.4
Risskov	36	215	6.1	**2400.4**	2427.4	1.1	2353.2	2.0	3.2
Roedkilde	18	230	4.4	2452.3	2534.1	3.3	**2495.7**	3.1	1.5
Rosborg1	22	257	4.9	2756.7	2895.6	5.0	**2837.4**	3.2	2.1
Rosborg2	22	257	4.8	2651.0	2860.6	7.9	**2805.4**	3.3	2.0
SanktAnnae1	23	149	3.6	1554.1	1675.1	7.8	**1580.4**	3.9	6.0
SanktAnnae2	24	165	3.8	1682.9	1850.2	9.9	**1753.4**	4.0	5.5
SanktAnnae3	17	21	2.6	197.9	197.9	0.0	197.9	0.0	0.0
SanktAnnae4	31	162	4.2	1359.0	1719.9	26.6	**1598.9**	3.6	7.6
Skanderborg1	57	232	3.9	2440.0	2640.1	8.2	**2547.4**	3.4	3.6
Skanderborg2	60	229	4.9	**2369.5**	2414.4	1.9	2320.4	3.4	4.1
Skive1	16	140	3.3	1417.5	1458.6	2.9	**1430.9**	3.8	1.9
Skive2	31	103	2.6	865.5	1062.6	22.8	**995.5**	2.5	6.8
Skive3	31	140	3.3	1189.7	1452.9	22.1	**1372.7**	4.4	5.8
Skive4	16	21	2.1	**227.8**	227.8	0.0	227.6	0.1	0.1
Skive5	16	98	3.0	**960.9**	972.9	1.3	960.7	0.9	1.3
Skive6	16	110	3.1	1106.9	1144.7	3.4	**1119.3**	2.0	2.3
Skive7	31	134	3.0	1152.9	1365.6	18.5	**1284.6**	3.2	6.3
Skive8	16	107	3.0	1006.8	1015.8	0.9	**1007.1**	0.9	0.9
Skive9	31	100	3.0	907.8	1034.9	14.0	**983.0**	4.6	5.3
Soenderborg1	22	234	3.7	2105.1	2590.6	23.1	**2475.3**	4.9	4.7
Soenderborg2	22	236	3.4	2095.1	2703.9	29.1	**2577.9**	4.2	4.9
Soenderborg3	21	236	3.4	2452.5	2688.1	9.6	**2597.3**	4.0	3.5
Soenderborg4	22	235	3.4	1927.6	2681.3	39.1	**2554.2**	3.7	5.0
Solroed1	18	242	4.6	1935.5	2181.4	12.7	**2130.3**	6.8	2.4
Solroed2	20	22	1.8	228.6	228.6	0.0	228.6	0.0	0.0
Solroed3	17	22	1.8	223.3	223.3	0.0	223.3	0.0	0.0
Solroed4	54	243	4.6	2309.9	2562.4	10.9	**2354.2**	5.2	8.8
Solroed5	20	243	4.5	1555.0	2185.8	40.6	**2054.9**	8.0	6.4
Solroed6	17	215	4.2	1703.0	1835.0	7.8	**1775.5**	6.8	3.4
Solroed7	17	194	4.0	1599.6	1748.0	9.3	**1679.2**	2.6	4.1
Vejen1	10	41	2.4	424.2	424.2	0.0	424.2	0.0	0.0
Vejen2	19	126	3.8	1171.2	1225.0	4.6	**1198.5**	3.0	2.2
Vejen3	19	125	4.2	1172.1	1234.9	5.4	**1204.6**	2.4	2.5
Vejen4	19	125	4.2	1130.2	1205.3	6.6	**1172.5**	2.4	2.8
Viborg1	19	105	3.8	1004.3	1100.3	9.6	**1034.2**	0.6	6.4
Viborg2	49	187	4.4	**2081.7**	2153.0	3.4	2060.0	4.8	4.5
Viby1	20	124	3.7	**1278.3**	1279.4	0.1	1256.9	1.2	1.8
Viby2	13	93	5.2	957.5	957.5	0.0	957.5	0.0	0.0
Viby3	8	45	2.7	480.0	480.0	0.0	480.0	0.0	0.0
Viby4	16	93	5.2	**1057.7**	1057.7	0.0	1053.3	0.3	0.4
Viby5	21	123	3.7	**1378.6**	1378.8	0.0	1356.1	1.1	1.7

Continued on next page

Table 11.4 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Average	30	148	4.1			7.7			4.1
Max	102	303	8.6			83.6			21.9

From Table 11.4 it is seen that ALNS in average finds solutions 4.1% from optimum for the SCTP. This is lower than the average gap for Gurobi, which is 7.7%.

From Table 11.3 it is seen that ALNS outperforms Gurobi for the PCTP. For the SCTP, the results are more blurred, but the ALNS still performs best in 70 out of 100 cases. What can also be seen from Table 11.3 and 11.4 is that the standard deviation for the ALNS is low in all cases, and the maximum gap obtained across all datasets is considerably lower than the maximum gap which Gurobi obtains (even given the higher running time of Gurobi). This is important as the customers of Lectio expects a consistent and stable solution procedure.

11.6.2 Performance comparison of ALNS and current heuristic of Lectio

The current algorithm in Lectio is an undocumented heuristic, which initially attempts to fulfill every meeting request by assigning them to random time slots, and then attempts to find improving solutions with a hill-climber embedded in a Simulated Annealing (SA) framework. This heuristic does not support the SCTP. In this section, we compare the existing heuristic of Lectio with the implemented ALNS algorithm.

The comparison of algorithms for the PCTP is done by adapting the objective of the ALNS so it matches the one of the implemented SA algorithm, which yields the following changes:

- Since the SA algorithm attempts to fulfill all meeting requests, we set α_g^s to a huge value for all meeting requests, effectively making the ALNS behave the same way.
- We set $\beta^t = \gamma^s = 2$.
- The SA algorithm does not allow interrupting of activities, i.e. $\delta^t = \delta^s = \infty$.
- The time slot set-point setting of the SA solver is broken, so we set $\kappa = 0$ and likewise for the SA solver.
- The violation of sequence length for teachers is penalized in quadratic way. This means that term (11.21) is now written as $-\sum_{t,b,d}(y_{t,b,d})^{\omega}$, and $\omega = 2$.

We evaluate the algorithms on 100 randomly selected datasets for the school-year 2009/2010. The reason a new batch of datasets is selected for this test is that the existing heuristic of Lectio does not support all features mentioned in this paper. Due to customer requests, Lectio is continuously developed, and this also effects the CTP. E.g. datasets from the school-year 2009/2010 does not support features such as multiple days for a consultation.

Experience has shown that the SA algorithm needs long runtime to provide meaningful solutions. We set runtime equal to 10 minutes, which is significantly higher than the preferable runtime of the high schools, as described in Section 11.6.1. Furthermore, to reduce the influence of stochastic behavior, we perform 10 runs on each dataset with each solver.

Table 11.5 shows the average performance, the standard deviation and the number of unassigned requests of both algorithms. Recall that in this test both algorithms attempt to fulfill

every meeting request. However it cannot be guaranteed that the algorithms can find a solution which satisfies this, nor that it even exists. Furthermore we compare the algorithms in the domain of the SA algorithm, and in this domain it is only attempted to minimize the different costs, i.e. no benefit is made from fulfilling meeting requests. This means it would not be fair to compare solutions which does not have the same amount of unassigned requests, since additional fulfilled requests will potentially yield additional cost of e.g. number of breaks, interrupted activities, etc. Therefore we enforce the following criterion to determine whether the comparison of solutions for a dataset is valid: The difference in the number of unassigned requests must lie in the interval ± 1. This means the comparison of solvers in Table 11.5 is only approximate, however it can be considered as a very good approximation, since a difference of one fulfilled meeting request will have minor influence of the objective. Notice that the objectives are given in the domain of the SA algorithm, which is of different magnitude than the objective of this article. This is due to the fact that the undocumented heuristic aims at minimizing whereas the approach of this paper is to maximize.

Given the average objective of the SA algorithm $\bar{x}_{\text{SA}}$ and the average objective of the ALNS algorithm $\bar{x}_{\text{ALNS}}$, and that the comparison of these two is valid, we compute the difference $\bar{x}_{\text{SA}} - \bar{x}_{\text{ALNS}}$. For almost every instance where comparison is valid, the ALNS algorithm in average finds a better solution. Furthermore the solutions from the ALNS algorithm has far lower deviation than the SA algorithm, which are important, as the users of the algorithm will usually only run the algorithm once.

Table 11.5: Comparison of performance of the SA algorithm and the ALNS algorithm. Each algorithm is ran 10 times on each dataset. For each algorithm and each dataset is listed the mean objective "$\bar{x}$", standard deviation of objective "σ", and the number of unassigned meeting requests "#UA". Notice that the objectives are given in the domain of the SA algorithm, which is of different magnitude than the objective of this article. Those datasets where the number of unassigned requests differs between the algorithms with more than ± 1 are struck out, as this is not considered a fair comparison. Column "Diff" is the difference between mean objectives.

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Aabenraa1	640.90	83.36	17.0	**555.00**	0.00	17.0	85.90
Aabenraa2	384.00	41.86	9.0	**315.90**	18.27	9.0	68.10
~~Aabenraa3~~	~~41.60~~	~~11.27~~	~~12.4~~	~~123.90~~	~~7.61~~	~~9.0~~	~~N/A~~
~~Aabenraa4~~	~~126.90~~	~~59.36~~	~~13.5~~	~~166.80~~	~~53.45~~	~~12.0~~	~~N/A~~
Aabenraa5	**36.00**	9.30	0.0	37.60	0.84	0.0	-1.60
Aabenraa6	0.00	0.00	60.0	0.00	0.00	60.0	0.00
Aabenraa7	0.00	0.00	52.0	0.00	0.00	52.0	0.00
Aurehoej	283.30	37.99	12.8	**165.70**	12.72	12.0	117.60
~~Birkeroed1~~	~~542.10~~	~~88.05~~	~~40.2~~	~~171.70~~	~~37.57~~	~~35.5~~	~~N/A~~
Birkeroed2	493.80	88.41	30.1	**92.40**	19.21	29.5	401.40
Bjerringbro	2994.60	390.30	22.0	**1393.20**	16.90	22.0	1601.40
~~Borupgaard1~~	~~36.60~~	~~11.66~~	~~10.9~~	~~176.70~~	~~11.98~~	~~2.0~~	~~N/A~~
~~Borupgaard2~~	~~16.00~~	~~8.03~~	~~51.2~~	~~221.40~~	~~1.71~~	~~41.0~~	~~N/A~~
~~Borupgaard3~~	~~32.80~~	~~23.67~~	~~26.3~~	~~115.80~~	~~10.34~~	~~20.0~~	~~N/A~~
Broenderslev	154.40	51.04	0.0	**22.20**	6.96	0.0	132.20
CPHWEST	1244.40	38.56	0.0	**1105.00**	0.00	0.0	139.40
Esbjerg	0.00	0.00	407.0	0.00	0.00	407.0	0.00

Continued on next page

Table 11.5 – continued from previous page

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Fjerritslev	2088.50	72.25	0.0	**1904.20**	6.34	0.0	184.30
Frederikssund1	330.40	39.98	0.0	**257.70**	6.34	0.0	72.70
Frederikssund2	436.40	22.85	0.0	**340.00**	4.59	0.0	96.40
Frederiksvaerk	25.90	1.85	34.0	**25.00**	0.00	34.0	0.90
GlHellerup1	130.90	29.37	4.0	**29.50**	5.78	4.0	101.40
GlHellerup2	268.70	36.58	1.0	**88.60**	12.95	1.0	180.10
Gefion	112.40	22.04	0.0	**16.70**	3.65	0.0	95.70
~~Gladsaxe~~	~~85.40~~	~~16.40~~	~~87.8~~	~~1034.00~~	~~37.54~~	~~27.0~~	~~N/A~~
Haderslev1	0.00	0.00	225.0	0.00	0.00	225.0	0.00
Haderslev2	0.00	0.00	185.0	0.00	0.00	185.0	0.00
Haslev1	1239.10	56.96	0.0	**1181.00**	0.00	0.0	58.10
Haslev2	0.00	0.00	51.0	0.00	0.00	51.0	0.00
Hasseris1	10.20	4.52	0.0	**0.90**	0.57	0.0	9.30
Hasseris2	68.80	18.94	0.0	**14.90**	2.56	0.0	53.90
Hasseris3	34.30	25.95	2.9	**5.10**	2.02	3.0	29.20
Hasseris4	11.10	6.23	0.0	**0.10**	0.32	0.0	11.00
Herlufsholm1	46.70	8.90	0.0	**34.30**	5.54	0.0	12.40
Herlufsholm2	86.30	14.35	1.2	**60.20**	4.05	0.0	26.10
~~Herlufsholm3~~	~~2.40~~	~~2.41~~	~~38.4~~	~~0.00~~	~~0.00~~	~~37.0~~	~~N/A~~
~~Herlufsholm4~~	~~267.30~~	~~32.27~~	~~3.3~~	~~176.90~~	~~9.80~~	~~2.0~~	~~N/A~~
Herlufsholm5	15.70	11.86	0.0	**1.60**	1.84	0.0	14.10
Herning1	843.30	119.31	1.0	**632.10**	9.00	1.0	211.20
Herning2	103.50	16.25	0.0	**74.20**	0.42	0.0	29.30
Himmelev	262.10	35.44	0.0	**96.30**	16.22	0.0	165.80
Horsens	17.20	7.50	2.0	**6.50**	1.35	2.0	10.70
Kongsholm	102.20	39.82	21.4	**62.30**	8.64	21.0	39.90
~~Langkaer~~	~~29.20~~	~~8.22~~	~~11.7~~	~~190.90~~	~~10.38~~	~~0.0~~	~~N/A~~
Mariagerfjord	190.00	52.75	0.9	**89.00**	17.58	1.0	101.00
Morsoe1	17.20	9.74	3.0	**0.50**	0.71	3.0	16.70
Morsoe2	44.00	21.91	0.0	**2.70**	1.42	0.0	41.30
Mulernes1	21.50	8.22	0.0	**2.60**	0.97	0.0	18.90
Mulernes2	27.60	12.51	0.0	**1.30**	1.16	0.0	26.30
NZahles1	97.40	28.46	2.2	**64.50**	0.71	2.0	32.90
NZahles2	111.20	35.32	0.9	**72.60**	6.10	0.0	38.60
NielsSteensen1	309.80	42.98	3.0	**214.40**	38.20	3.0	95.40
NielsSteensen2	124.10	24.92	0.0	**90.50**	22.65	0.0	33.60
NielsSteensen3	186.40	37.65	5.0	**147.60**	22.31	5.0	38.80
Nordsjaelland1	1036.30	74.52	4.0	**652.20**	55.28	4.0	384.10
Nordsjaelland2	69.50	2.17	586.0	**68.50**	1.58	586.0	1.00
Nordsjaelland3	**1681.40**	132.07	45.0	1927.20	191.16	45.0	-245.80
Nordsjaelland4	**1627.80**	135.70	65.7	2335.90	270.87	65.8	-708.10
Nordsjaelland5	1699.10	118.08	27.6	**1104.40**	76.82	27.0	594.70
Nordsjaelland6	1369.00	102.13	32.1	**1109.90**	61.44	32.0	259.10
Nordsjaelland7	1646.40	106.74	49.1	**1567.70**	87.35	49.0	78.70
Nordsjaelland8	2065.30	179.20	37.0	**1736.00**	93.59	37.0	329.30

Continued on next page

Table 11.5 – continued from previous page

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Nyborg1	89.70	16.45	0.0	**72.10**	0.32	0.0	17.60
Nyborg2	93.50	23.75	0.0	**51.00**	1.49	0.0	42.50
Nyborg3	0.40	0.70	55.6	**0.00**	0.00	55.0	0.40
Nyborg4	14.40	6.75	3.0	**1.50**	0.85	3.0	12.90
Naerum	424.50	43.07	9.0	**248.40**	50.63	9.0	176.10
Noerresundby1	36.10	18.11	0.0	**9.90**	3.54	0.0	26.20
Noerresundby2	115.70	26.60	0.3	**34.90**	10.12	0.0	80.80
Odder	69.80	27.68	6.0	**21.00**	0.00	6.0	48.80
~~Oure~~	~~40.80~~	~~11.13~~	~~15.7~~	~~24.40~~	~~8.22~~	~~14.0~~	~~N/A~~
Paderup	141.50	15.81	0.0	**75.50**	4.48	0.0	66.00
Randers1	168.40	22.17	0.0	**93.50**	0.53	0.0	74.90
Randers2	244.00	77.90	0.0	**10.80**	3.79	0.0	233.20
Rosborg1	0.00	0.00	246.0	0.00	0.00	246.0	0.00
~~Rosborg2~~	~~2611.00~~	~~299.16~~	~~19.1~~	~~837.10~~	~~75.08~~	~~18.0~~	~~N/A~~
Roskilde	202.30	60.81	0.0	**139.00**	23.25	0.0	63.30
Rybners1	274.90	45.42	0.0	**174.90**	1.10	0.0	100.00
Rybners2	1043.80	67.33	15.0	**789.10**	4.15	15.0	254.70
Roedkilde1	10.90	4.84	0.0	**2.00**	1.33	0.0	8.90
Roedkilde2	19.30	7.48	7.0	**12.00**	2.67	7.0	7.30
Roedkilde3	57.10	11.75	0.0	**6.80**	2.35	0.0	50.30
~~SanktAnnae1~~	~~205.90~~	~~57.43~~	~~60.3~~	~~201.30~~	~~22.07~~	~~51.6~~	~~N/A~~
SanktAnnae2	0.00	0.00	7.0	0.00	0.00	7.0	0.00
Skanderborg1	3.50	1.65	0.0	**0.00**	0.00	0.0	3.50
Skanderborg2	35.40	34.59	0.9	**5.80**	3.79	1.0	29.60
Skive	176.50	47.52	18.0	**89.00**	25.05	18.0	87.50
Slagelse1	130.80	5.94	3.0	**114.90**	5.49	3.0	15.90
Slagelse2	29.70	19.44	0.2	**3.50**	2.55	0.0	26.20
Soroe	2117.10	111.78	14.5	**883.20**	49.02	13.5	1233.90
Stoevring1	17.50	4.93	0.0	**2.70**	1.42	0.0	14.80
Stoevring2	229.20	55.04	0.0	**16.40**	4.14	0.0	212.80
Stoevring3	260.60	71.41	0.0	**14.30**	2.36	0.0	246.30
Stoevring4	124.70	48.48	0.0	**13.80**	2.25	0.0	110.90
Stoevring5	65.90	22.83	6.0	**10.30**	2.54	6.0	55.60
Varde	867.40	152.80	10.0	**326.20**	9.46	10.0	541.20
Vejen	0.30	0.48	0.0	**0.00**	0.00	0.0	0.30
ViborgG	267.90	8.84	94.0	**257.80**	1.03	94.0	10.10
ViborgH	102.20	28.61	0.0	**22.60**	9.16	0.0	79.60
~~Viby~~	~~236.30~~	~~3.65~~	~~186.8~~	~~230.60~~	~~0.52~~	~~181.8~~	~~N/A~~
Average	431.53	47.78	33.9	319.50	20.14	32.5	133.60

By the computational tests of this section, it has been shown that the ALNS algorithm is the best solution procedure, of those considered in this paper, for the CTP. It outperforms both Gurobi and the existing heuristic of Lectio in terms of both solution quality and reliability.

11.7 Final Remarks and Outlook

It has been shown how the CTP, an important real-life problem for the Danish high schools, can be modeled using linear IP. ALNS has proven successful in establishing solutions for two versions of the problem, the PCTP and the SCTP. Furthermore, F-Race has shown to be an efficient method for tuning of the free parameters. The developed ALNS algorithm has been implemented in Lectio and is hence available for 95% of the Danish high schools.

In case of the PCTP, it has been shown that the ALNS algorithm in average finds solutions which are less than 4% from optimum. This average is taken over 100 real-life dataset, and therefore we have high confidence in this result. Furthermore it has been shown that comparing with the existing algorithm in Lectio, which is the only other known heuristic algorithm for the problem, the ALNS algorithm is far superior. For 83 of the 86 datasets, ALNS finds better solutions, and in many cases the solution quality of the ALNS is considerably better. For the remaining 14 datasets a comparison was not considered fair.

The performance for the SCTP is also tested on 100 real-life dataset. For these datasets, it is shown that the ALNS algorithm in average finds solutions less than 5% from optimum.

For both the PCTP and SCTP the average solution found by ALNS is better than the solutions found by the state-of-the-art MIP solver Gurobi 5.01.

The main subject for further research is considered to be the use of Dantzig-Wolfe decomposition and solution using Branch-and-Price. In this context, a column in the master problem could represent a meeting-plan for a student or a teacher. This would move many constraints to the subproblem, possibly giving a stronger IP formulation, which could lead to a more efficient IP-based solution approach.

Another possibility for future research is to combine the two solution methods described, i.e. using the MIP solver as a repair heuristic within the ALNS. Similar approaches are seen in Muller et al. (2011) and Prescott-Gagnon et al. (2009), with competitive results.

Bibliography

B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips*. CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

S. Becker, J. Gottlieb, and T. Stützle. Applications of racing algorithms: An industrial perspective. In E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, volume 3871 of *Lecture Notes in Computer Science*, pages 271–283. Springer Berlin / Heidelberg, 2006.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9:403–432, 2006. ISSN 1094-6136.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

Y. Diao, F. Eskesen, S. Froehlich, J. Hellerstein, L. Spainhower, and M. Surendra. Generic online optimization of multiple configuration parameters with application to a database server. In M. Brunner and A. Keller, editors, *Self-Managing Distributed Systems*, volume 2867 of *Lecture Notes in Computer Science*, pages 79–93. Springer Berlin / Heidelberg, 2003.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36:267–306, September 2009. ISSN 1076-9757.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775 – 1783, 2011. ISSN 0305-0548. doi: DOI:10.1016/j.cor.2011.02.007.

B. McCollum. University timetabling: Bridging the gap between research and practice. In *in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 15–35. Springer, 2006.

H. Mittelman. Benchmarks for optimization software, 2013. URL `http://plato.asu.edu/bench.html`.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

E. Montero, M.-C. Riff, and B. Neveu. An evaluation of off-line calibration techniques for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 299–300, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0072-8. doi: 10.1145/1830483.1830540.

L. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

L. F. Muller and S. Spoorendonk. A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem. Technical report, DTU Management Engineering, 2010.

P. Pellegrini and M. Birattari. Implementation effort and performance. pages 31–45. 2007.

P. Pellegrini, T. Stützle, and M. Birattari. Off-line vs on-line tuning: A study on max–min ant system for the tsp. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-15460-7.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54 (4):190–204, 2009. ISSN 1097-0037. doi: 10.1002/net.20332.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

M. Sørensen and T. R. Stidsen. High school timetabling: Modeling and solving a large number of cases in denmark. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 359–364. SINTEF, 2012.

A. Tripathy. School timetabling–a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, 1984.

Chapter 12 Paper G

A Branch & Price Algorithm for the Generalized Meeting Planning Problem

Niels-Christian Fink Bagger[1], Matias Sørensen[1,2], Simon Kristiansen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract Meetings form an important activity in modern society. Persons meet with other people for business or pleasure, but usually a meeting rules out other meetings for the same persons at the same time. Planning a large number of meetings hence present an important but difficult optimization problem.

The Meeting Planning problem comes in all kinds of different variations. In this article, we present a Mixed-Integer Programming model which can model many of these variations and we also present a Branch & Price algorithm which enables optimization of many types of meeting problems. The Branch & Price algorithm is tested on two problems occurring in Danish high schools, the Parental Consultation Timetabling Problem and the Supervisor Consultation Timetabling Problem, each represented by 100 real-world datasets. The tests show that the Branch & Price algorithm performs well compared with a compact Mixed-Integer Programming model and with an Adaptive Large Neighborhood Search algorithm.

12.1 Introduction

Meetings form an important activity in modern society, both in people's business- and private life. If the occurrence of meetings is large, and the meetings are interrelated, planning them becomes a puzzle. In this article we will study the *Generalized Meeting Planning problem*.

Every day numerous secretaries have to solve the following planning problem: Find a meeting time for a number of persons to meet. If the secretary is using a standard electronic calendar system, there usually exists a function for finding the first time-interval where all persons are available. This is a simple function and very useful for the meeting planners. If the secretary has to plan more meetings, the function works as well, unless one or more of the persons have to participate in several meetings. Then planning one meeting renders these persons unavailable to participate in other meetings in the same time-interval, if we assume that a person can only participate in one meeting at a time.

The problem of planning a number of meetings in time-intervals (time slots), some of which requires the participation of the same person, is the Generalized Meeting Planning (GMP) prob-

lem. We need to stress that the term *meeting* should be understood in its most general form: A meeting consists of a number of *entities* meeting each other in a specific time-interval. The entities can be managers, teachers and students, or any other type of resource attending the meetings.

There are naturally many variations of the GMP problem, both in the objective, i.e. what should be achieved, and in the constraints, i.e. which extra requirements are there. On the other hand, all GMPs share the same basic structure and a planning approach, i.e. optimization, for one version of a GMP may benefit from approaches for other GMPs.

The Mixed-Integer Programming (MIP) model that we will present makes it obvious to apply a Branch & Price (B&P) algorithm to find the optimal solution, due to an exponential number of variables. This works by splitting the model in a *master problem* and a number of *subproblems*. The Linear Programming (LP) relaxation of the master problem, denoted the Relaxed Master Problem (RMP), will be solved using a Column Generation (CG) algorithm. To find integer solutions by means of this LP relaxation, a Branch & Bound scheme is used, which thereby leads to a B&P algorithm. B&P algorithms have been very successful for a large number of hard optimization problems, e.g. Savelsbergh (1997); Mehrotra and Trick (2007). For an excellent introduction to B&P we refer to Lübbecke and Desrosiers (2005).

This paper is based on a technical report Bagger (2012), in which a number of different approaches to the GMP problem are attempted. We will refer to this technical report for details on various matters throughout this paper.

The outline of the paper is as follows. In Section 12.2 we will briefly survey previous approaches. In Section 12.3 we define the GMP problem as a MIP model. We will test our approach on two different variations of the GMP problem which are described in Section 12.4: The Parental Consultations Timetabling problem in Section 12.4.1, and the Supervisor Consultation Timetabling Problem in Section 12.4.2. The problems are formulated more generally as the Consultation Timetabling Problem. In both tests, we will use real-life data from a large number of Danish high schools. Computational results are presented in Section 12.5. Finally we will give a conclusion in Section 12.6.

12.2 Previous Approaches

The generalized version of the meeting planning problem considered is widely applicable, as the required structure is simple; A number of meetings between entities (e.g. individuals) are to be scheduled, and each entity can participate in multiple meetings. Hence both *entity* and *meeting* are actually used as abstracts terms here, since their definitions are problem-specific. To the best of our knowledge, the abstract form of this problem has not been considered before in the literature. However, several well-described problem domains exist for which the GMP is applicable, especially within timetabling problems in the educational sector.

In case of *High School Timetabling*, a meeting could be defined as a *lecture*, and entities could be students, teachers, classes, or a combination of these. Santos et al. (2012) describes a Cut and Column Generation algorithm for class/teacher timetabling at Brazilian high schools, where the set of teachers is used as entities. Papoutsis et al. (2003) also uses teachers and entities, and schedules for Greek high schools.

The field of *University Timetabling* is dominated by heuristic approaches. The problem is closely related to graph coloring and many early approaches were based on graph coloring heuristics, but recently meta-heuristics have gained the most interest Lewis (2008). However some researchers have applied column generation approaches. Qualizza and Serafini (2005) describes an approach where a column represents the assigning of time slots for a single course. Computational

results for a single problem instance shows convincing results.

For *Examination Planning* many different approaches have been suggested, though most heuristic. A survey by Qu et. al in Qu et al. (2006) considers the recent research applied in the field. Here it is noted that decomposition methods have not attracted much attention. One of the reasons is that some soft constraints cannot be evaluated in the decomposition and so global optimality may be missed. Like for university timetabling, most of the early approaches applied graph-heuristics, but lately methods such as Tabu Search Pais and Amaral (2008); Di Gaspero and Schaerf (2001) and Simulated Annealing Thompson and Dowsland (1996b,a) have gained attention. In Thompson and Dowsland (1996a) it is mentioned that this method resulted in an implementation which were used at University of Wales Swansea. Defining a single exam as a meeting between students (usually only a single student) and a group of teachers would allow applying the terminology of this paper.

In this paper we consider the case of the *Consultation Timetabling Problem* for high schools in Denmark, which essentially consists of meetings between a group of teachers and a single student. This problem is only described in the literature in Kristiansen et al. (2013).

12.3 A Mixed-Integer Programming model of the Generalized Meeting Planning problem

There are numerous ways to model the GMP problem using MIP models. In Bagger (2012) a number of these are presented, including a compact model (i.e. a model with a polynomially limited number of variables and constraints). Here we will only present what is known as the *Entity Pattern model*, as this performed best wrt. computational results.

An *entity* is a resource that has to participate in a number of meetings. Let the set of entities which are part of the GMP problem be given by the set E, indexed by $e \in E$. Let further the set of time slots be given by B, indexed by $b \in B$. Each meeting concerns a group g of entities which should meet and which is part of a set of groups G, i.e. $g \in G$.

Given the three sets E, B and G, further data about the problem is given: $\alpha_{g,b}$ is the profit of scheduling a meeting $g \in G$ to time slot $b \in B$. The incidence matrix A_g^e defines the groups such that $A_g^e = 1$ if entity $e \in E$ belongs to group $g \in G$ and $A_g^e = 0$ otherwise.

In the Entity Pattern model we will use two types of variables: The binary variable $x_{g,b}$ which defines that group $g \in G$ meets in time slot $b \in B$, and the binary variable $\lambda^{e,p}$ which defines whether the meetings of entity $e \in E$ follows the patterns $p \in P$. A pattern is essentially a schedule for an entity, which defines the time slots in which the entity is attending a meeting, but does not explicit denote which meetings. The number of patterns for an entity may be exponential. We will here assume to know the entire set of patterns P_e for each entity, and we will also assume to know the cost $\beta^{e,p}$ for using entity pattern $p \in P_e$. Let incidence matrix $M_b^{e,p}$ take value 1 if pattern $p \in P_e$ does *not* allow a meeting in time slot $b \in B$.

The entire Entity Pattern model can now be formulated, see Model 1. This constitutes the master problem in our CG formulation. The notation used is lazy; Let $\forall a$ be shorthand for $\forall a \in A$, and let $\sum_a$ be shorthand for $\sum_{a \in A}$.

Model 1 consists of one objective function and three types of constraints: The objective function given by eq. (1a) weighs the two terms, the revenue for allocating a meeting and the revenue of the entity pattern for each entity. What exactly is contained in the revenue term of the groups α and the revenue term of the patterns β is dependent on the actual meeting planning problem. In the settings we consider, different properties of the patterns are penalized, e.g. too many consecutive meetings, idle time slots and so on. It is desired to minimize these penalties, so we let β take non-positive values. The constraint in eq. (1b) ensures that each meeting can be

Model 1 Master problem of the pattern formulation.

$$\max \sum_{g,b} \alpha_{g,b} \cdot x_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p} \tag{1a}$$

$$\text{s.t.} \sum_{b} x_{g,b} \leq 1 \quad \forall g \tag{1b}$$

$$\sum_{g} A_g^e \cdot x_{g,b} + \sum_{p \in P_e} \lambda^{e,p} \cdot M_b^{e,p} = 1 \quad \forall e, b \tag{1c}$$

$$\sum_{p \in P_e} \lambda^{e,p} = 1 \quad \forall e \tag{1d}$$

$$x_{g,b}, \lambda^{e,p} \in \{0,1\} \tag{1e}$$

arranged at most one time. The constraint in eq. (1c) links the group allocation variables $x_{g,b}$ with the entity patterns $\lambda^{e,p}$, i.e. ensures that an entity can only be assigned a time slot which is allowed. Finally the convexity constraint given by eq. (1d) ensures that exactly one entity pattern is chosen for each entity.

Of all the approaches attempted in Bagger (2012), Model 1 performed best in the tests and it indeed possess a number of features which makes it attractive:

1. Model 1 can be converted into a set-partition model. These models are very well studied which means that the theory is well-developed and a number of good performing heuristics are known.

2. The entity patterns can be generated in the subproblem when using a CG algorithm, hence overcoming the problem of the exponential number of $\lambda^{e,p}$ variables.

3. Relations between meetings for one entity in the entity pattern is handled in the subproblem of the CG algorithm.

4. The $x_{g,b}$ variables can be used in a B&P algorithm, such that branching is performed only on these variables. A proof of this will be given in Section 12.3.2.

5. Branching only on the $x_{g,b}$ variables does not change the structure of the subproblem, as opposed to other branching schemes, facilitating a simpler B&P algorithm.

Because of the above reasons, we consider the entity pattern Model 1 a promising model. There are however a couple of weaknesses which also deserves to be mentioned:

1. Any relation between entity patterns, i.e. between the meetings of different entities becomes more cumbersome to model.

2. There is no definition of where the meetings should take place, e.g. allocation of meetings to rooms. For simplicity we have chosen to ignore this issue in this paper.

12.3.1 Obtaining Dual Bounds

When solving the RMP problem a CG algorithm will be used. Using this approach only a subset of the patterns are initially included in the model. In this case all the $x_{g,b}$ variables are initially

included in the RMP and one $\lambda^{e,p}$ variable for each entity derived from an initial solution. This problem is called the Restricted Master Problem and the dual information of the solution is used to generate new variables (columns) to include in the model. When using CG to solve the RMP a dual bound can be calculated in each iteration. Let υ_g, $\pi_{e,b}$ and μ_e be the dual variables of the constraints (1b), (1c) and (1d) respectively. Consider some iteration of the CG procedure and let $(\overline{\upsilon}_g, \overline{\pi}^e_b, \overline{\mu}_e)$ be the dual solution of the Restricted Master Problem. Then the subproblem consists of finding the pattern with the highest reduced cost for each entity. For an entity e the subproblem can be denoted in the following way:

$$z^e_{\mathrm{SP}}\left(\overline{\pi}, \overline{\mu}\right) = \max\left\{ \beta^{e,p} - \sum_b \overline{\pi}^e_b \cdot M^{e,p}_b - \overline{\mu}_e \,\middle|\, p \in P_e \right\} \tag{12.1}$$

Let $\left(x^*_{g,b}, \lambda^{e,p,*}\right)$ be the optimal solution to the RMP and let $\left(\overline{x}_{g,b}, \overline{\lambda}^{e,p}, \overline{\upsilon}_g, \overline{\pi}^e_b, \overline{\mu}_e\right)$ be the primal-dual solution to the Restricted Master Problem in any iteration. Furthermore let the optimal solution for the subproblem for entity $e \in E$ given the dual solution be denoted $\overline{z}^e_{\mathrm{SP}}\left(\overline{\pi}, \overline{\mu}\right)$. Lastly because of the strong duality theorem we have that:

$$\sum_{g,b} \alpha_{g,b} \cdot \overline{x}_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \overline{\lambda}^{e,p} = \sum_g \overline{\upsilon}_g + \sum_{e,b} \overline{\pi}^e_b + \sum_e \overline{\mu}_e \tag{12.2}$$

So a bound on the gap between the optimal solution to the RMP and the current considered solution in the Restricted Master Problem can be calculated:

$$\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*} - \sum_g \overline{\upsilon}_g - \sum_{e,b} \overline{\pi}^e_b - \sum_e \overline{\mu}_e \leq \tag{12.3}$$

$$\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*}$$

$$- \sum_{g,b} \overline{\upsilon}_g \cdot x^*_{g,b} - \sum_{e,b} \overline{\pi}^e_b \cdot \left(\sum_g A^e_g \cdot x^*_{g,b} + \sum_{p \in P_e} \lambda^{e,p,*} \cdot M^{e,p}_b \right) - \sum_{e,p \in P_e} \overline{\mu}_e \cdot \lambda^{e,p,*} = \tag{12.4}$$

$$\sum_{g,b} \left(\alpha_{g,b} - \overline{\upsilon}_g - \sum_e A^e_g \cdot \overline{\pi}^e_b \right) \cdot x^*_{g,b} + \sum_{e,p \in P_e} \left(\beta^{e,p} - \sum_b \overline{\upsilon}^e_b \cdot M^{e,p}_b - \overline{\mu}_e \right) \cdot \lambda^{e,p,*} \tag{12.5}$$

The step from eq. (12.3) to eq. (12.4) is done by using eq. (1b), (1c) and (1d) since they must hold for the optimal solution. The step from eq. (12.4) to eq. (12.5) is merely a rearrangement of the terms.

If the RMP is dualized then, for a group g and a time slot b, one of the constraints that are obtained is $\upsilon_g + \sum_e A^e_g \cdot \pi^e_b \leq \alpha_{g,b}$. Since $x_{g,b}$ are non-negative variables then this means that $\sum_{g,b} \left(\alpha_{g,b} - \overline{\upsilon}_g - \sum_e A^e_g \cdot \overline{\pi}^e_b\right) \cdot x^*_{g,b} \leq 0$ thus removing this term from (12.5) provides an upper bound. Now consider the coefficient of the $\lambda^{e,p,*}$-variables in (12.5). These correspond to the objective function of the subproblems in the current iteration of the CG algorithm. Since $\overline{z}^e_{\mathrm{SP}}(\overline{\pi}, \overline{\mu})$ is the objective value of the optimal pattern for the subproblem of entity e then each coefficient must be bounded by this value, i.e.

$$\sum_{e,p \in P_e} \left(\beta^{e,p} - \sum_b \overline{\upsilon}^e_b \cdot M^{e,p}_b - \overline{\mu}^e \right) \cdot \lambda^{e,p,*} \leq \sum_e \overline{z}^e_{\mathrm{SP}}(\overline{\pi}, \overline{w}) \cdot \left(\sum_{p \in P_e} \lambda^{e,p,*} \right) \tag{12.6}$$

As (1d) constrains $\sum_{p \in P_e} \lambda^{e,p,*}$ to be one for each entity e then we can replace this sum by one and so an upper bound on the optimal solution of the RMP can be calculated:

$$\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*} \leq \sum_{g,b} \alpha_{g,b} \cdot \overline{x}_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \overline{\lambda}^{e,p} + \sum_{e} \overline{z}^e_{\mathrm{SP}}(\overline{\pi}, \overline{\mu}) \quad (12.7)$$

This means that if all the subproblems are solved to optimality then we can obtain a dual bound for the RMP in every iteration of the CG algorithm at a negligible computational cost. So in a B&P algorithm this upper bound can be checked against the incumbent in each iteration of the CG algorithm. If the bound gets below the incumbent in some iteration, the node can be fathomed before all the columns are generated instead of putting off the fathoming until the node is solved to optimality. The drawback of this approach is that we need to generate the optimal columns in each iteration to get the correct bound. However this can be circumvented by replacing $\overline{z}^e_{\mathrm{SP}}$ with an upper bound on the subproblem (12.1). In our testing we will only add the optimal solutions of the subproblem.

12.3.2 Branching in the Branch & Price Algorithm

Model 1 has the important feature that it is possible to only branch on the $x_{g,b}$ variables. This simplifies the algorithm significantly because the standard B&P branching techniques can be avoided. In the remainder of this section we will show that branching on only the $x_{g,b}$ variables is indeed enough to obtain the optimal integer solution, i.e. that branching on the $\lambda^{e,p}$ variables is never necessary. To show this we will use the properties of *perfect matrices* Ryan and Falkner (1988).

Definition 12.1. (Padberg (1974); Ryan and Falkner (1988)) *An $m \times n$ zero-one set-partitioning polyhedron characterized by*
$\{x | x \in \mathbb{R}^n, Ax = e, x \geq 0\}$ $\left(\text{where } e = (1, 1, \ldots, 1)^T\right)$ *is said to be perfect if it has only zero-one integral vertices.*

Assume that in some node in the Branch & Bound tree the optimal solution to the RMP is integral on the $x_{g,b}$ variables and that all the branching performed until this node has only been done on the $x_{g,b}$ variables. Let $\overline{x}_{g,b}$ be the solution and assume that all feasible patterns for the entities are in the model. If the $x_{g,b}$ variables are substituted with the values $\overline{x}_{g,b}$ in Model 1 then the remaining problem will be as in Model 2.

Model 2 Master problem of the pattern formulation given the integral solution $\overline{x}_{g,b}$.

$$\max \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p} \quad (2a)$$

$$\text{s.t.} \sum_{p \in P_e} \lambda^{e,p} \cdot M^{e,p}_b = 1 - \sum_g A^e_g \cdot \overline{x}_{g,b} \quad \forall e, b \quad (2b)$$

$$\sum_{p \in P_e} \lambda^{e,p} = 1 \quad \forall e \quad (2c)$$

$$\lambda^{e,p} \in \{0, 1\} \quad (2d)$$

Consider any entity e. If $\sum_g A^e_g \cdot \overline{x}_{g,b} = 1$ for any time slot b then the corresponding constraint and every pattern $p \in P_e$ where $M^{e,p}_b = 1$ can be removed from the model. Removing all the

constraints and patterns fulfilling the latter creates a zero-one set partitioning problem. Note that the model is decomposable by the entities, i.e. the optimal solution for an entity is independent of the remaining entities. The problem is very easy to solve since each entity can only be assigned one pattern and each of the patterns can be deduced from the $\overline{x}_{g,b}$ values. However since the solution is based on the RMP it could be the case that fractional patterns were chosen and further branching were needed on the $\lambda^{e,p}$ variables.

To see if there exists an integral solution to the problem in Model 2, it could be checked whether the set-partitioning polyhedron is perfect. However this might not be so easy to show but, as mentioned earlier, since the problem is decomposable on the entities then subproblems can be generated by decomposing Model 2 on the entities. If the set partitioning polyhedron of each of these subproblems are perfect then the subproblems has integral optimal solutions meaning that the overall problem has an integral optimal solution. To see that the subproblems are perfect we need Definition 12.2 and Theorem 10.

Definition 12.2. (Padberg (1974); Ryan and Falkner (1988)) *An $m \times k$ zero-one matrix A_k with $k \leq m$ is said to have the property $\Pi_{\beta,k}$ if:*

- *A_k contains a $k \times k$ non-singular submatrix B_k where all rows and columns sum to β.*
- *Every row in A_k which is not in B_k is either equal to a row in B_k or the sum of the row is strictly less than β.*

Theorem 10. (Padberg (1974); Ryan and Falkner (1988)) *A perfect $m \times n$ zero-one matrix A does not contain any $m \times k$ submatrix A_k where $3 \leq k \leq m$ with the property $\Pi_{\beta,k}$ where $\beta \geq 2$.*

Consider the subproblem for any entity and let A be the corresponding zero-one matrix which could look like the following:

$$A = \begin{bmatrix} 1 & 1 & & 1 & \cdots & 1 \\ & 1 & 1 & & \cdots & \\ 1 & & & 1 & \cdots & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & 1 & & 1 & \cdots & 1 \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

Note the last row in the matrix which corresponds to the convexity constraint for the entity and thus has ones in all the columns. This row ensures that the matrix does not contain a submatrix A_k with property $\Pi_{\beta,k}$.

Claim 12.3. *A is a perfect matrix.*

Proof of Claim 12.3. To prove that A is a perfect matrix we will show by contradiction that A fulfills Theorem 10. Assume that there is a submatrix A_k of A with property $\Pi_{\beta,k}$ and let B_k be the corresponding $k \times k$ non-singular submatrix of A_k where all rows and columns sum to β. Consider the last row r corresponding to the convexity constraint. We will give the following two claims:

Claim 12.4. *Row r cannot be one of the rows in the submatrix B_k.*

Claim 12.5. *Row r cannot be one of the remaining rows of A_k not in B_k.*

Assuming that both Claim 12.4 and 12.5 are true then clearly they present a contradiction thus Claim 12.3 must hold. This concludes that A is a perfect matrix. ∎

Proof of Claim 12.4. We know that row r sums to k in the matrix A_k since it has ones in all columns. This means that if r is in B_k then all the other rows must also sum to k which can only be done if they have ones in all columns as well. This contradicts that B_k is non-singular and so r cannot be one of the rows in B_k. ∎

Proof of Claim 12.5. All the rows in B_k must sum to β which must strictly be less than k, otherwise B_k would not be non-singular as mentioned in the proof of Claim 12.4. Since row r sums to k then this contradicts the assumption that A_k had property $\Pi_{\beta,k}$ as this would imply that the sum of row r should be strictly less than $\beta < k$. Therefore the row r cannot be one of the rows in A_k which is not in B_k. ∎

Since Claim 12.3 holds then if the $x_{g,b}$ variables are integral in the optimal solution for the RMP in some node of the B&P tree then there exists an integral optimal solution for the RMP in that node. The patterns that can be deduced from the $x_{g,b}$ variables are the only patterns that can be used in an integral solution because of the convexity constraints and so these patterns must be the optimal patterns to choose. This means that the branching can be done solely on the $x_{g,b}$ variables, i.e. the subproblems do not need to be changed throughout the branching tree.

12.3.3 Branching Selection

When branching on a fractional $x_{g,b}$ variable, one problem is that the one-branch creates a more restricted subproblem than the zero-branch. Therefore the B&P tree becomes very unbalanced. This is a well-known issue. A way to circumvent this could for instance be to use constraint branching as the Ryan-Foster branching method for which empirical studies has shown that this approach performs well on set partitioning problems Ryan and Foster (1981). The branching is performed by identifying two constraints r and s. Either these two constraints must be fulfilled by the same variable or by different variables. So if a_{rj} and $a_{s,j}$ are the coefficients associated with the variable x_j in the constraints r and s respectively then the branching is done in the following way:

$$\sum_{j:a_{rj}=a_{sj}=1} x_j = 0 \quad \vee \quad \sum_{j:a_{rj}=a_{sj}=1} x_j = 1$$

The difference between the Ryan-Foster branching and variable branching in this context only occurs if two entities are in more than one group together. We cannot guarantee that this is true, and for the tests performed in Section 12.4 this is rarely the case, so the majority of the Ryan-Foster branching will result in branching on a single variable. Therefore we have decided to do a variable branching instead.

So the choice resides on how to do the variable branching. A usual approach is to branch on a variable x which is *most fractional*, i.e. a variable where the fractional part is closest to 0.5. Another idea is to consider a subset of all the fractional variables as potential candidates for branching, and then for each of these variables solve the relaxations of the two subproblems. This branching scheme is known as *strong branching* and the candidates we will consider are the k candidates which are *closest* to 0.65, as done by Røpke (Røpke, 2013). Instead of just solving the relaxations of all the subproblems induced by the candidates, a more clever approach to speed up strong branching is used, as described in Røpke (2012).

The main difference between our implementation and that of Røpke is the choice of the candidates to consider. The k candidates which are closest to 0.65 are chosen to be considered when branching. Røpke chose to set $k = 30$ in the beginning of the B&P algorithm and then at

some point lower k to 15 Røpke (2013). The reason for this is that it is usually more important to do good branching decisions in the beginning of the B&P algorithm. Instead of choosing a depth of the tree or a number of processed nodes to be the breaking point of when to lower k, we have chosen to determine k based on the number of fixed variables. This is because when a variable is fixed to zero then in most cases this is a very weak restriction and so two subproblems at the same depth in the tree can have a great difference in the size of the problem in terms of non-fixed variables. One thing to note is that when one of the $x_{g,b}$ is fixed to one then a lot of other variables get fixed to zero. So when we calculate the number of fixed variables we will count every $|B| - 1$ variables which are fixed to zero as one variable fixed to one. This means that if in some node of the B&P tree $\mathcal{O}$ is the set of variables fixed to one and $\mathcal{Z}$ is the set of variables fixed to zero then the *counted* variable fixings is $|\mathcal{O}| + |\mathcal{Z}|/(|B| - 1)$. Then for every tenth *counted* variable fixing, we half the value of k beginning with 60.

12.4 Test Applications

In this section we describe two different applications of the GMP problem. Both of these applications originate from the Danish high school system, and the description of the problems given in the following has shown to be applicable to hundreds of different high schools. Sections 12.4.1 and 12.4.2 describe the two test-applications, and Section 12.4.3 presents a subproblem in context of GMP which can be used to solve both applications. In Kristiansen et al. (2013) it is shown that both test-applications are $\mathcal{NP}$-hard and since these are a special case of GMP this means that GMP is $\mathcal{NP}$-hard as well.

12.4.1 Parental Consultation Timetabling Problem

The first application is the *Parental Consultation Timetabling Problem* (PCTP). Few times a year, the Danish high schools, 9th to 12th school year, offer the possibility for school meetings where the parents together with the pupil meet selected teachers for short private meetings. The pupil and the parents will choose a number of teachers they wish to meet. At a certain date, the booking possibility is closed, and a secretary at the high school will use an optimization algorithm (see Kristiansen et al. (2013)) to plan the meetings. At the day of the meeting, the parents and pupil will come to the school, and the different teachers will be spread out in different class-rooms at the school. The parents and the pupil are then given a schedule, for which teacher to meet in which time slot. The parental meetings are often placed in the evening of a normal workday to ease the attendance of the parents. Evening work is however more demanding for the teachers, so compact schedules becomes important. The GMP problem can be used to find the best schedules of both the parents and the teachers.

12.4.2 Supervisor Consultation Timetabling Problem

The second application is for *Supervisor Consultation Timetabling Problem* (SCTP). In the final year for a student at a Danish high school, the student is required to do a large study project. Each student selects two course subjects and thereby two teachers whom will be supervisors on the project. During the project process, the students are required to attend a meeting with their supervisors. These meetings are used to give the students some guidance for different parts for the project, such as problem definition or literature research.

The planning of the SCTP is very similar to the PCTP, however in SCTP the students have only one request for a meeting (where both supervisors attend). These meetings are normally placed in the daytime, hence it is often necessary to interrupt the lectures for a given student.

However it is not allowed to assign a meeting during lunch breaks. As each student only has one meeting the SCTP essentially consists of finding the best schedules for the teachers.

12.4.3 Generalized Meeting Planning Subproblem for the Consultation Timetabling Problem

There are a number of requirements to be fulfilled for an entity pattern to be feasible for both PCTP and SCTP:

- For entity $e \in E$ the incidence matrix $C_b^e \in \{0,1\}$ defines whether a meeting can take place in time slot b. A teacher might be unavailable if he is occupied by other activities, such as teaching.
- For each entity $e \in E$ only a maximum number of meetings Q_e in a row can be accepted. This can either be a hard or soft constraint specified by $HS_e \in \{0,1\}$. If the constraint is hard ($HS_e = 1$) then a meeting cannot be scheduled after a sequence of Q_e meetings. If the constraint is soft ($HS_e = 0$) then a sequence can be longer than Q_e but at a cost of ω_e times the number of time slots that the sequence exceeds Q_e.

We introduce the set of days, D, indexed by $d \in D$. The parameter $V_{b,d} \in \{0,1\}$ denotes whether time slot b is at day d. Let υ_g, $\pi_{e,b}$ and μ_e be the dual variables associated with the constraints (1b), (1c) and (1d), respectively. Let $C_b^e \in \{0,1\}$ take value 1 if entity e is available in time slot b, and 0 otherwise. Let $m_b \in \{1 - C_b^e, 1\}$ be a variable denoting whether the entity does *not* have a meeting in time slot b. Let $f_d^{\text{first}} \in \{0,1\}$ and $f_d^{\text{last}} \in \{0,1\}$ indicate the first and last time slot the entity has a meeting in day $d \in D$, respectively.

The cost function for idle time slots are modeled as a piece-wise linear function by introducing the variable $v_{d,j} \in \{0,1\}$, where $j \in \{1,\ldots,m\}$, which takes value 1 if entity e has j idle time slots in day d, and 0 otherwise. The variable $y_{b,d} \in \mathbb{R}^+$ takes value 1 if the length of the sequence starting in time slot b on day d exceeds Q_e. Let variable $u_d \in \{0,1\}$ take value 1 if the entity has at least one meeting at day d, and 0 otherwise, and variable $v \in \mathbb{R}^+$ denotes the amount of days the entity has scheduled meetings, minus one.

Model 3 shows the entire subproblem.

Constraints (3b) and (3c) are not essentially needed to get feasible patterns, however they can help avoiding infeasible solutions. As an example say entity e_1 is in two groups and that the dual values make it attractive for entity e_1 to have meetings in time slots 1 and 6. Assigning meetings for e_1 in these two time slots would create a lot of idle time slots in the schedule and therefore assigning meetings in time slot 2, 3, 4 and 5 (assuming these time slots are feasible for the entity) would remove the idle time slots. This is an infeasible pattern as e_1 is in two groups but scheduled for six meetings. Constraints (3b) ensure that only requested meeting are assigned and (3c) ensure that a group can only be placed at time slot b if that time slot is available for the entity. Constraints (3f), (3g) and (3h) denote the number of idle time slots the entity has at a given day. If an idle time slot is required after each sequence of meetings of size Q_e in (3h) (i.e. HS = 1) this is not counted as a idle time slot. Constraints (3i) are convexity constraints ensuring that only one of the idle time slot variables is set to one.

Constraints (3d) and (3e) are the hard sequential constraints. If $HS_e = 1$, a sequence of meetings cannot exceed Q_e. Constraints (3j) are the soft constraints of sequential meetings (i.e. $HS_e = 0$). The way the number of time slots that exceeds Q_e in some sequence of length k is computed by settings lower bounds of 1 on the first $k - Q_e$ of the $y_{b,d}$ variables. As an example let an optimal solution for the entity have scheduled meeting in the time slots 1, 3, 4, 5, 6, 7, 8 and 10 which are all connected to day 1 and let $Q_e = 4$ then the sequence 3, 4, 5, 6, 7, 8

Model 3 subproblem of the entity patterns formulation given the dual values $\{\overline{v}_{e,b}\}$ and $\{\overline{w}_e\}$.

$$\max(e, \overline{\pi}, \overline{\mu}) - \sum_{d,j} \gamma^e_{d,j} \cdot v_{d,j} - \sum_{b,d} \omega_e \cdot y_{b,d} - \zeta_e \cdot v - \sum_b \overline{\pi}_{e,b} \cdot m_b - \overline{\mu}_e \tag{3a}$$

$$\text{s.t.} \sum_b x_{g,b} \leq A^e_g \quad \forall g \tag{3b}$$

$$\sum_g A^e_g \cdot x_{g,b} + m_b = 1 \quad \forall b \tag{3c}$$

$$\sum_g \sum_{b'=b}^{b+Q_{e'}} V_{b',d} \cdot A^{e'}_g \cdot x_{g,b'} \leq Q_{e'} \quad \begin{array}{l}\forall d, b \leq |B| - Q_{e'} - 1, \\ e' \neq e, HS_{e'} = 1, V_{b,d} = 1\end{array} \tag{3d}$$

$$\sum_{b'=b}^{b+Q_e} V_{b',d} \cdot m_{b'} \geq 1 \quad \begin{array}{l}\forall d, b \leq |B| - Q_e - 1, \\ HS_e = 1, V_{b,d} = 1\end{array} \tag{3e}$$

$$f^{\text{last}}_d + b \cdot m_b \geq b \quad \forall b, d, V_{b,d} = 1 \tag{3f}$$

$$f^{\text{first}}_d - (|B| - b - 1) \cdot m_b \leq b \quad \forall b, d, V_{b,d} = 1 \tag{3g}$$

$$f^{\text{last}}_d - f^{\text{first}}_d + \left(1 + \frac{HS_e}{Q_e}\right) \cdot \sum_b V_{b,d} \cdot (1 - m_b) - \sum_j j \cdot v_{d,j} \leq -\frac{HS_e}{Q_e} \quad \forall d \tag{3h}$$

$$\sum_j v_{d,j} = 1 \quad \forall d \tag{3i}$$

$$\sum_{b'=b}^{b+Q_e} V_{b',d} \cdot m_{b'} + y_{b,d} \geq 1 \quad \begin{array}{l}\forall d, b \leq |B| - Q_e - 1, \\ HS_e = 0, V_{b,d} = 1\end{array} \tag{3j}$$

$$m_b + u_d \geq 1 \quad \forall b, d, V_{b,d} = 1 \tag{3k}$$

$$\sum_d u_d - v \leq 1 \tag{3l}$$

$$x_{g,b}, v_{d,j} \in \{0, 1\} \tag{3m}$$

$$m_b \in \{1 - C^e_b, 1\} \tag{3n}$$

$$f_{\text{first}}, f_{\text{last}}, y_{b,d}, u_d, v \in \mathbb{R}^+ \tag{3o}$$

exceeds Q_e by 2 meetings. This means that $y_{3,1}$ and $y_{4,1}$ both will get a lower bound of 1 and since they are not bounded by any other constraint they will be set to 1 in the optimal solution. This means that this sequence will be correctly penalized by 2 times ω_e in the objective function. Constraints (3k) and (3l) sets the values for the variables u_d and v.

12.5 Computational Results

To evaluate the B&P algorithm we compare it with two other known solution approaches for both the PCTP and the SCTP. These are described in detail in Kristiansen et al. (2013) as well as the values of the different parameters of the model.

- *Adaptive Large Neighborhood Search*: This is a heuristic based on Adaptive Large Neighborhood Search, which is currently used by many high schools in Denmark to solve both the PCTP and the SCTP.
- *Gurobi 5.0.1*: This denotes the performance of Gurobi v. 5.0.1 on a standard MIP model (the formulation from Bagger (2012) where Gurobi performed best). The results reported here are not the same ones used in Kristiansen et al. (2013) as new runs were performed

to obtain more information. Therefore small variations are seen in the data.

The implementation was done in C# 4.5 running on a Windows machine equipped with an Intel i7 CPU clocked at 2.80GHz and with 12GB of RAM. Gurobi 5.0.1 was used as LP solver for the master problem and as MIP solver for the subproblems (with default parameter settings). The maximum running time for Gurobi and B&P was set to 1 hour, while the ALNS ran 10 times with a time limit of 2 minutes (we report the average objective values). Thereby the comparison is 'unfair' in favor of Gurobi and B&P, which should be kept in mind throughout this section.

As a starting solution to the B&P algorithm we use a solution obtained by a single run of the ALNS algorithm. Since the ALNS algorithm is stochastic, the B&P algorithm is also stochastic, but for time reasons only a single run of the B&P algorithm was performed. It should be noted that the deviation between solutions obtained by ALNS for the same datasets have experimentally been shown to be low. Therefore the solutions obtained by the B&P algorithm are also expected to have low deviation between them. The single run of ALNS to get the initial solution is included in the reported running time for B&P.

All datasets have been obtained from the database of the commercial product Lectio, which is used by hundreds of high schools in Denmark. Thereby all datasets represent a case of a real-world optimization problem.

12.5.1 Parental Consultation Timetabling Problem

Table 12.1 shows the results obtained for the PCTP. Table 12.2 shows these results in summarized format.

Table 12.1: Computational results for 100 datasets for the PCTP problem. Column 'ALNS' denotes the solution obtained by a problem-specific *Adaptive Large Neighborhood Search* heuristic. For Gurobi the obtained lower- and upper bound (columns 'Obj' and 'UB' respectively) is shown, as well as the final gap. The amount of required seconds is also shown. For the B&P algorithm, the final number of explored nodes in the B&P tree is shown, as well as the obtained bounds, the final gap, and total run time. Note that a node is only counted as being explored after a branching has been performed, i.e. if the optimal solution is found in the root node the number of explored nodes is 0. The best found solution is marked in **bold**, and the best found bound is marked with a '*'.

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Alleroed	51	12	38	484.8	**485.0**	*485.0	0.0	3	67	**485.0**	*485.0	0.0	167
Alssund	84	18	55	849.4	**850.6**	*850.6	0.0	503	112	850.4	850.8	0.1	>3600
Aurehoej1	537	18	189	3655.9	3429.4	3773.0	10.0	>3600	148	**3660.7**	*3763.0	2.8	>3600
Aurehoej2	409	18	170	**3219.5**	2976.6	3297.1	10.8	>3600	154	3219.3	*3272.0	1.6	>3600
Broenderslev	241	24	111	1905.6	1650.4	1965.2	19.1	>3600	88	**1910.9**	*1936.0	1.3	>3600
CPHWEST	133	39	59	**1044.5**	1001.8	1124.0	12.2	>3600	32	1041.4	*1067.8	2.5	>3600
DetKristne	247	32	83	**1830.9**	1455.6	1973.1	35.6	>3600	40	1829.5	*1900.2	3.9	>3600
Dronninglund1	108	30	36	782.0	744.0	801.5	7.7	>3600	210	**783.9**	*798.0	1.8	>3600
Dronninglund2	94	30	34	**664.5**	641.4	672.5	4.9	>3600	810	658.2	*671.8	2.1	>3600
Egaa	265	24	109	**2318.5**	2132.1	2374.0	11.4	>3600	140	2317.7	*2365.0	2.0	>3600
Egedal	408	27	186	3558.2	3091.5	3625.2	17.3	>3600	16	**3568.3**	*3608.4	1.1	>3600
Esbjerg1	345	24	124	2402.1	2333.5	2466.1	5.7	>3600	228	**2403.5**	*2440.3	1.5	>3600
Esbjerg2	307	24	160	2314.3	2119.4	2440.3	15.1	>3600	92	**2315.6**	*2365.7	2.2	>3600
Esbjerg3	255	24	94	**1839.5**	1801.8	1885.6	4.7	>3600	32	1837.8	*1860.1	1.2	>3600
Esbjerg4	351	24	126	2612.3	2451.6	2706.2	10.4	>3600	228	**2616.7**	*2657.2	1.5	>3600
Frederikssund	49	24	26	403.8	**404.2**	406.6	0.6	>3600	161	**404.2**	*404.2	0.0	1513
Frederiksvaerk	74	8	54	697.9	**699.0**	*699.0	0.0	22	168	**699.0**	*699.0	0.0	186
Gefion	479	18	220	**3958.1**	3309.4	4248.8	28.4	>3600	68	3920.6	*4175.6	6.5	>3600
Gladsaxe	901	40	302	**6950.7**	5443.7	7163.9	31.6	>3600	0	6944.2	*7142.7	2.9	>3600
Greve	336	18	152	**2482.6**	2192.9	2535.9	15.6	>3600	78	2474.3	*2532.6	2.4	>3600
Haslev1	123	18	65	1060.2	1058.0	1072.6	1.4	>3600	115	**1061.3**	*1061.3	0.0	475

Continued on next page

Table 12.1 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Haslev2	122	18	67	988.7	977.2	1020.9	4.5	>3600	1512	**989.8**	*1004.4	1.5	>3600
Herlufsholm1	143	24	43	894.5	842.2	919.1	9.1	>3600	533	**895.1**	*905.1	1.1	>3600
Herlufsholm2	88	24	31	621.8	594.8	664.4	11.7	>3600	720	**623.6**	*633.1	1.5	>3600
Herning1	118	27	64	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Herning2	75	27	34	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Herning3	140	27	69	**2.0**	**2.0**	*2.0	0.0	0	0	**2.0**	*2.0	0.0	120
Himmelev	453	34	177	3471.7	3093.4	3735.0	20.7	>3600	10	**3483.9**	*3586.7	2.9	>3600
Hjoerring	179	30	132	**1009.2**	790.1	1231.4	55.9	>3600	0	999.2	*1178.3	17.9	>3600
HorsensGym	123	18	75	1129.4	**1132.4**	*1132.5	0.0	2544	467	1129.3	1133.8	0.4	>3600
HorsensStats	143	21	98	1253.8	1217.8	1286.3	5.6	>3600	802	**1254.2**	*1268.1	1.1	>3600
Ikast-Brande	52	30	24	**449.7**	448.5	458.6	2.3	>3600	169	**449.7**	*452.7	0.7	>3600
Johannesskolen1	165	24	58	**1188.3**	1136.3	1293.1	13.8	>3600	496	1187.1	*1212.3	2.1	>3600
Johannesskolen2	97	24	39	743.9	704.1	786.4	11.7	>3600	788	**748.0**	*761.7	1.8	>3600
Johannesskolen3	135	28	56	519.9	500.3	588.4	17.6	>3600	508	**522.0**	*548.2	5.0	>3600
Kalundborg	299	27	164	**2458.3**	2212.2	2592.0	17.2	>3600	66	2457.0	*2517.8	2.5	>3600
Kolding	80	18	43	721.0	**721.4**	723.2	0.2	>3600	998	721.1	*721.8	0.1	>3600
Langkær1	52	18	35	470.6	**471.1**	*471.1	0.0	309	269	**471.1**	*471.1	0.0	750
Langkær2	90	18	50	805.9	788.6	814.1	3.2	>3600	945	**806.0**	*810.0	0.5	>3600
Middelfart	223	27	109	**1788.1**	1653.9	1916.6	15.9	>3600	98	1784.6	*1821.5	2.1	>3600
Morsoe1	105	27	37	**804.2**	755.1	834.1	10.5	>3600	352	802.3	*824.2	2.7	>3600
Morsoe2	113	27	35	**778.4**	715.1	815.0	14.0	>3600	300	773.0	*806.5	4.3	>3600
Munkensdam1	256	21	127	2198.6	1851.6	2252.1	21.6	>3600	74	**2210.4**	*2234.8	1.1	>3600
Munkensdam2	345	21	157	**2846.8**	2464.9	2930.1	18.9	>3600	42	2831.1	*2905.2	2.6	>3600
NielsSteensens1	117	36	35	757.2	704.3	781.8	11.0	>3600	38	**757.4**	*776.9	2.6	>3600
NielsSteensens2	328	30	63	**1656.6**	1414.6	1763.7	24.7	>3600	52	1655.3	*1728.0	4.4	>3600
NielsSteensens3	365	30	64	1800.3	1587.7	1895.6	19.4	>3600	50	**1800.7**	*1859.4	3.3	>3600
NielsSteensens4	234	30	52	1144.4	1086.9	1229.4	13.1	>3600	4	**1146.3**	*1191.0	3.9	>3600
NielsSteensens5	263	30	57	1557.1	1451.1	1632.9	12.5	>3600	110	**1560.5**	*1606.9	3.0	>3600
Noerre	422	18	209	3944.5	3469.9	4033.6	16.3	>3600	100	**3952.3**	*4011.7	1.5	>3600
Nordfyns	192	23	102	**1795.5**	1782.3	1858.8	4.3	>3600	412	1793.4	*1836.9	2.4	>3600
Nordsjaellands1	1187	34	232	**6597.3**	5867.3	7020.9	19.7	>3600	6	6565.1	*6971.6	6.2	>3600
Nordsjaellands2	1038	34	203	2453.2	2300.1	2624.8	14.1	>3600	50	**2460.0**	*2582.7	5.0	>3600
Nordsjaellands3	457	34	106	**2634.7**	2092.3	2858.0	36.6	>3600	8	2633.4	*2759.4	4.8	>3600
Nordsjaellands4	163	34	50	1172.4	1117.5	1209.8	8.3	>3600	92	**1173.0**	*1206.1	2.8	>3600
Nordsjaellands5	712	40	164	4460.5	3808.0	4801.9	26.1	>3600	4	**4467.5**	*4692.6	5.0	>3600
Nordsjaellands6	780	34	170	**4612.2**	4025.4	4897.2	21.7	>3600	16	4599.6	*4829.8	5.0	>3600
Nordsjaellands7	880	34	190	**2894.5**	2527.6	3048.3	20.6	>3600	22	2884.9	*3010.3	4.3	>3600
Nordsjaellands8	23	34	21	241.9	**242.1**	*242.1	0.0	6	21	**242.1**	*242.1	0.0	261
Nordsjaellands9	949	34	196	5037.1	4479.7	5518.7	23.2	>3600	12	**5056.7**	*5426.4	7.3	>3600
Nordsjaellands10	31	34	23	269.1	270.4	276.1	2.1	>3600	58	**272.2**	*272.2	0.0	479
Nyborg	119	24	58	**55.4**	**55.4**	*55.4	0.0	0	0	**55.4**	*55.4	0.0	120
Nykoebing	182	24	118	1483.1	1471.3	1495.2	1.6	>3600	661	**1484.0**	*1489.3	0.4	>3600
NZahles1	324	25	109	**2365.7**	1936.3	2456.2	26.9	>3600	44	2356.3	*2447.8	3.9	>3600
NZahles2	301	24	107	**2217.8**	2000.0	2280.1	14.0	>3600	68	2211.6	*2275.6	2.9	>3600
Odder	95	18	37	**762.7**	751.8	773.2	2.8	>3600	832	761.8	*772.6	1.4	>3600
Odsherreds	193	21	96	1595.4	1548.3	1619.7	4.6	>3600	24	**1597.5**	*1610.0	0.8	>3600
Risskov1	65	15	35	536.7	**539.7**	*539.7	0.0	577	212	**539.7**	*539.7	0.0	387
Risskov2	149	15	69	1256.9	**1264.6**	1272.0	0.6	>3600	1624	1248.6	*1270.2	1.7	>3600
Risskov3	181	15	78	1389.7	**1402.7**	*1406.5	0.3	>3600	1920	1391.1	1407.3	1.2	>3600
Roedkilde	266	18	103	**2325.2**	2218.2	2352.9	6.1	>3600	32	2321.1	*2352.7	1.4	>3600
Roedovre	779	51	291	1661.7	1482.8	2008.2	35.4	>3600	14	**1683.7**	*1812.1	7.6	>3600
Rosborg1	218	24	85	**1827.9**	1752.1	1876.2	7.1	>3600	522	1798.5	*1866.6	3.8	>3600
Rosborg2	268	28	95	2223.3	1972.9	2297.3	16.4	>3600	154	**2229.4**	*2273.5	2.0	>3600
Rosborg3	487	28	307	4750.0	4602.1	4938.2	7.3	>3600	124	**4752.9**	*4824.9	1.5	>3600
Rosborg4	235	26	94	1960.4	1643.4	2033.0	23.7	>3600	184	**1967.7**	*2015.7	2.4	>3600
Roskilde	263	48	113	**2112.8**	1663.0	2252.6	35.5	>3600	0	2107.7	*2171.1	3.0	>3600
Rybners	267	24	126	**2402.5**	1952.4	2473.4	26.7	>3600	34	2395.1	*2446.5	2.1	>3600
SanktAnnae	320	21	124	**2381.6**	2114.8	2499.1	18.2	>3600	48	2375.6	*2458.5	3.5	>3600
Skive	220	36	78	**1850.3**	1604.5	1903.2	18.6	>3600	34	1849.0	*1877.5	1.5	>3600
Slagelse	85	30	50	802.6	**805.3**	*805.3	0.0	337	164	802.3	805.4	0.4	>3600
Solroed1	341	16	148	**2436.8**	2397.5	2468.9	3.0	>3600	378	2426.7	*2468.3	1.7	>3600
Solroed2	415	16	182	**3263.0**	3140.2	3318.4	5.7	>3600	238	3256.1	*3315.8	1.8	>3600
Soroe1	369	24	145	2947.3	2648.0	3106.8	17.3	>3600	58	**2957.9**	*3042.5	2.9	>3600
Soroe2	335	33	139	**2255.0**	1615.9	2641.1	63.5	>3600	10	2247.6	*2441.3	8.6	>3600
Stenhus	221	18	121	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Stoevring	62	24	31	520.5	**521.4**	*521.4	0.0	21	76	**521.4**	*521.4	0.0	335
Struer1	237	30	127	**1656.9**	1610.0	1794.2	11.4	>3600	160	1656.4	*1697.2	2.5	>3600
Struer2	333	30	138	**2534.0**	2094.3	2793.9	33.4	>3600	104	2532.5	*2637.4	4.1	>3600
Svendborg1	96	18	79	991.1	**991.4**	*991.4	0.0	35	1619	**991.4**	*991.4	0.0	>3600

Continued on next page

Table 12.1 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Svendborg2	134	18	82	1288.0	**1289.1**	*1289.2	0.0	142	764	1288.2	1289.4	0.1	>3600
Taarnby	791	36	244	5609.2	4587.1	*5918.9	29.0	>3600	0	**5622.2**	5927.8	5.4	>3600
UCH	104	32	110	**922.6**	**922.6**	†*922.6	0.0	4	0	**922.6**	*922.6	0.0	143
ViborgGym1	206	30	88	1434.0	1348.6	1482.5	9.9	>3600	122	**1437.9**	*1467.2	2.0	>3600
ViborgGym2	149	30	62	1133.4	1101.2	1146.9	4.2	>3600	384	**1134.4**	*1138.1	0.3	>3600
ViborgGym3	294	30	143	**2211.7**	2081.2	2275.6	9.3	>3600	118	2210.0	*2236.0	1.2	>3600
ViborgHandel	324	30	95	2526.7	2160.1	2615.9	21.1	>3600	58	**2537.4**	*2614.2	3.0	>3600
ViborgKatedral	337	40	101	**2313.4**	1755.6	2516.6	43.4	>3600	8	2304.7	*2465.0	7.0	>3600
Vordingborg1	315	16	132	**2304.3**	2201.4	2353.5	6.9	>3600	38	2304.2	*2349.2	2.0	>3600
Vordingborg2	239	16	115	1924.6	1893.2	1949.4	3.0	>3600	436	**1926.9**	*1949.0	1.1	>3600

† A wrong upper bound was reported in Kristiansen et al. (2013).

Table 12.2: Summary of results for PCTP. 'Best obj' denotes the amount of instances where the algorithm provided the best objective value (including draws). 'Best UB' denotes the amount of instances where the algorithm found the best upper bound (including draws). Columns 'Gap $\leq q$ shows the amount of instances for which the respective algorithm provided a gap $\leq q$. 'Avg. Gap to best UB' is found for each algorithm by finding the best available UB for each instance, calculating the gap to the solution provided, and averaging these gaps.

	Best obj	Best UB	Gap = 0%	Gap ≤ 2%	Gap ≤ 5%	Avg. Gap to best UB
ALNS	46	-	-	-	-	2.31%
Gurobi	21	19	17	23	35	9.37%
B&P	54	94	16	54	92	2.32%

The most prominent numbers drawn from these results are the gap to the best upper bound. This shows that the B&P algorithm in average is only 2.32% within optimum, which is slightly worse than the ALNS algorithm, but far better than the solutions Gurobi provides. Furthermore the B&P algorithm finds solutions within 5% from optimum for 92 instances.

12.5.2 Supervisor Consultation Timetabling Problem

Table 12.3: Computational results for 100 datasets for the SCTP. Columns have same meaning as in Table 12.1.

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Aabenraa	226	60	279	**2387.5**	2111.1	†2490.8	18.0	>3600	0	2384.9	*2465.1	3.4	>3600
Broendby1	69	21	87	**683.2**	672.1	704.9	4.9	>3600	825	682.2	*689.7	1.1	>3600
Broendby2	69	14	85	768.6	770.9	779.4	1.1	>3600	2401	**772.0**	*772.4	0.0	>3600
Broendby3	62	24	80	614.8	603.9	633.7	4.9	>3600	669	**617.7**	*620.7	0.5	>3600
Broenderslev1	115	69	148	**1302.4**	1276.0	†*1341.7	5.2	>3600	0	1300.7	1347.2	3.6	>3600
Broenderslev2	115	102	148	**1236.0**	1142.4	†*1271.8	11.3	>3600	0	1234.9	1443.5	16.9	>3600
Christianshavns	210	43	254	**2223.7**	1723.3	†2386.1	38.5	>3600	44	2217.2	*2264.6	2.1	>3600
Dronninglund1	134	100	166	1453.5	1382.4	†*1480.7	7.1	>3600	0	**1454.7**	1668.1	14.7	>3600
Dronninglund2	134	60	165	1537.5	1530.2	†*1561.7	2.1	>3600	0	**1539.8**	1568.8	1.9	>3600
Egaa	214	29	259	2376.2	2228.1	2458.3	10.3	>3600	88	**2380.4**	*2408.5	1.2	>3600
Falkoner1	64	30	83	668.0	**671.6**	677.3	0.9	>3600	36	**671.6**	*671.6	0.0	216
Falkoner2	206	37	255	**2266.1**	2056.3	†2345.0	14.0	>3600	46	2265.2	*2297.4	1.4	>3600
Falkoner3	64	30	83	**664.9**	**664.9**	670.4	0.8	>3600	178	**664.9**	*664.9	0.0	554
Grenaa1	122	28	154	1325.4	1291.8	1379.6	6.8	>3600	164	**1328.5**	*1348.1	1.5	>3600
Grenaa2	122	24	154	1290.3	1264.9	1329.6	5.1	>3600	361	**1295.9**	*1306.7	0.8	>3600
Greve1	157	28	205	1693.3	1620.6	1761.3	8.7	>3600	428	**1695.9**	*1710.0	0.8	>3600

Continued on next page

Table 12.3 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Greve2	259	62	322	2913.5	2627.9	†3035.3	15.5	>3600	0	**2915.1**	*3025.2	3.8	>3600
Greve3	51	20	67	566.3	**566.4**	570.7	0.8	>3600	8	**566.4**	*566.4	0.0	138
Gribskov1	182	74	226	1787.6	1269.7	†1912.1	50.6	>3600	0	**1788.9**	*1849.1	3.4	>3600
Herlev1	71	24	96	730.2	728.3	750.7	3.1	>3600	980	**730.7**	*734.0	0.4	>3600
Herlev2	78	29	106	750.2	729.5	792.7	8.7	>3600	442	**750.7**	*764.8	1.9	>3600
Hoeng1	66	21	85	621.5	605.6	688.0	13.6	>3600	4	**621.8**	*621.8	0.0	139
Hoeng2	98	23	123	1038.5	1036.0	1070.2	3.3	>3600	84	**1041.4**	*1041.4	0.0	298
Hoeng3	45	22	62	408.3	387.0	479.9	24.0	>3600	20	**409.8**	*409.8	0.0	167
Hoeng4	56	23	78	589.2	590.0	612.4	3.8	>3600	368	**591.9**	*591.9	0.0	721
Koebenhavns1	143	16	180	1242.0	**1246.7**	1273.4	2.1	>3600	1406	1243.5	*1256.4	1.0	>3600
Koebenhavns2	100	16	127	785.6	**786.4**	*786.5	0.0	197	2852	784.9	786.9	0.2	>3600
Koebenhavns3	100	16	127	**725.8**	**725.8**	*725.8	0.0	23	2672	**725.8**	*725.8	0.0	2991
Koebenhavns4	146	25	184	1424.6	1406.0	1486.1	5.7	>3600	842	**1424.7**	*1432.5	0.5	>3600
Koege1	255	30	285	2333.2	1039.0	2474.3	138.1	>3600	416	**2357.1**	*2388.2	1.3	>3600
Koege2	261	36	303	2045.7	1790.3	†2278.1	27.2	>3600	304	**2056.8**	*2075.2	0.9	>3600
Koege3	258	74	288	**2622.9**	2381.7	†2890.0	21.3	>3600	0	2621.3	*2809.1	7.2	>3600
Kolding1	219	24	263	2348.7	2287.1	2422.6	5.9	>3600	666	**2354.3**	*2364.0	0.4	>3600
Kolding2	174	45	220	1908.8	1727.2	†2003.2	16.0	>3600	28	**1910.9**	*1936.4	1.3	>3600
Langkaer1	215	62	255	**2239.4**	1661.7	†2465.4	48.4	>3600	0	2230.2	*2307.5	3.5	>3600
Langkaer2	216	60	256	**2240.1**	1746.1	†2477.1	41.9	>3600	0	2239.6	*2303.6	2.9	>3600
Langkaer3	216	60	256	2258.3	1950.4	†2470.4	26.7	>3600	0	**2267.4**	*2303.7	1.6	>3600
Langkaer4	57	30	72	566.2	562.5	594.5	5.7	>3600	398	**573.4**	*573.4	0.0	1128
Langkaer5	217	56	256	**2253.3**	1982.9	†2494.2	25.8	>3600	6	2252.3	*2299.1	2.1	>3600
Langkaer6	56	62	71	**629.2**	623.3	†652.0	4.6	>3600	128	629.1	*632.1	0.5	>3600
Mariagerfjord1	123	29	154	**1318.6**	1216.6	1385.9	13.9	>3600	266	1315.9	*1331.2	1.2	>3600
Mariagerfjord2	123	29	154	1345.3	1282.9	1401.3	9.2	>3600	172	**1347.6**	*1362.7	1.1	>3600
Marselisborg1	102	22	132	**1045.4**	1019.3	1090.3	7.0	>3600	1130	1044.5	*1053.4	0.8	>3600
Marselisborg2	106	17	138	1035.4	1035.4	1047.3	1.2	>3600	694	**1037.1**	*1037.1	0.0	995
Marselisborg3	105	22	132	1098.3	1076.3	1155.4	7.4	>3600	126	**1106.0**	*1106.0	0.0	524
Marselisborg4	96	17	126	947.2	**948.0**	952.9	0.5	>3600	327	947.7	*951.2	0.4	>3600
Munkensdam	191	43	225	2067.9	1741.7	†2201.2	26.4	>3600	14	**2075.6**	*2111.2	1.7	>3600
Noerresundby	303	31	367	3291.7	3131.4	3460.7	10.5	>3600	132	**3304.5**	*3320.1	0.5	>3600
Nordfyns1	173	22	207	1926.6	1884.0	1972.4	4.7	>3600	398	**1930.3**	*1940.3	0.5	>3600
Nordfyns2	173	21	206	**1929.9**	1905.6	1975.0	3.6	>3600	488	1929.4	*1945.9	0.9	>3600
Nordfyns3	173	22	207	1908.1	1870.5	1972.9	5.5	>3600	464	**1911.8**	*1924.5	0.7	>3600
Nordfyns4	173	21	215	1478.3	1452.7	1536.9	5.8	>3600	486	**1478.6**	*1499.8	1.4	>3600
NZahles1	69	13	90	619.1	615.6	634.8	3.1	>3600	4625	**620.1**	*620.5	0.1	>3600
NZahles2	62	13	78	509.5	511.7	522.4	2.1	>3600	5453	**513.0**	*513.1	0.0	>3600
Odsherreds	119	49	150	1289.4	1211.3	†1367.2	12.9	>3600	24	**1291.0**	*1305.7	1.1	>3600
Oeregaard1	219	20	258	2258.1	2257.3	2295.7	1.7	>3600	492	**2259.3**	*2289.9	1.4	>3600
Oeregaard2	213	20	256	1743.9	1749.4	1810.2	3.5	>3600	362	**1774.0**	*1798.0	1.4	>3600
Oeregaard3	219	20	258	2340.2	2311.5	2372.4	2.6	>3600	338	**2343.2**	*2361.5	0.8	>3600
Oeregaard4	219	20	258	**2339.9**	2325.0	2371.7	2.0	>3600	378	2334.1	*2363.1	1.2	>3600
Risskov	215	36	250	**2353.2**	2165.8	†2426.8	12.1	>3600	26	2352.6	*2389.7	1.6	>3600
Roedkilde	230	18	282	**2495.7**	2473.3	2534.6	2.5	>3600	790	**2495.7**	*2523.0	1.1	>3600
Rosborg1	257	22	310	2837.4	2787.5	2895.3	3.9	>3600	182	**2839.7**	*2893.3	1.9	>3600
Rosborg2	257	22	311	2805.4	2640.8	2859.4	8.3	>3600	182	**2805.5**	*2838.8	1.2	>3600
SanktAnnae1	149	23	191	1580.4	1539.7	1671.3	8.6	>3600	462	**1584.5**	*1599.6	1.0	>3600
SanktAnnae2	165	24	209	1753.4	1689.4	1844.9	9.2	>3600	250	**1754.5**	*1773.3	1.1	>3600
SanktAnnae3	21	17	29	**197.9**	**197.9**	*197.9	0.0	7	45	**197.9**	*197.9	0.0	142
SanktAnnae4	162	31	201	**1598.9**	1415.3	1718.0	21.4	>3600	108	1593.4	*1634.7	2.6	>3600
Skanderborg1	232	57	291	**2547.4**	2131.4	†2640.6	23.9	>3600	0	2545.4	*2588.6	1.7	>3600
Skanderborg2	229	60	276	2320.4	2000.8	†2414.4	20.7	>3600	0	**2324.7**	*2374.6	2.2	>3600
Skive1	140	16	182	1430.9	1420.9	1459.2	2.7	>3600	1088	**1436.1**	*1444.9	0.6	>3600
Skive2	103	31	143	995.5	856.3	1061.3	24.0	>3600	126	**996.5**	*1025.7	2.9	>3600
Skive3	140	31	182	1372.7	1307.0	1451.9	11.1	>3600	144	**1373.1**	*1401.6	2.1	>3600
Skive4	21	16	31	227.6	**227.8**	*227.8	0.0	12	24	**227.8**	*227.8	0.0	141
Skive5	98	16	131	960.7	**963.6**	971.7	0.8	>3600	234	959.1	*965.4	0.7	>3600
Skive6	110	16	145	**1119.3**	1111.6	1143.1	2.8	>3600	298	1115.3	*1131.1	1.4	>3600
Skive7	134	31	179	1284.6	1169.3	1365.8	16.8	>3600	114	**1289.7**	*1310.0	1.6	>3600
Skive8	107	16	143	**1007.1**	1005.7	1016.5	1.1	>3600	2423	1006.9	*1008.9	0.2	>3600
Skive9	100	31	133	**983.0**	959.0	1034.9	7.9	>3600	358	979.1	*1000.7	2.2	>3600
Soenderborg1	234	22	298	**2475.3**	2262.0	2590.4	14.5	>3600	104	2456.3	*2515.5	2.4	>3600
Soenderborg2	236	22	305	**2577.9**	2297.0	2701.0	17.6	>3600	118	2569.9	*2617.3	1.8	>3600
Soenderborg3	236	21	305	2597.3	2288.3	2686.6	17.4	>3600	222	**2601.5**	*2619.2	0.7	>3600
Soenderborg4	235	22	304	2554.2	2341.7	2679.2	14.4	>3600	102	**2563.3**	*2597.2	1.3	>3600
Solroed1	242	18	295	2130.3	1983.7	2180.6	9.9	>3600	646	**2132.7**	*2164.6	1.5	>3600

Continued on next page

Table 12.3 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Solroed2	22	20	34	**228.6**	**228.6**	*228.6	0.0	0	0	**228.6**	*228.6	0.0	120
Solroed3	22	17	34	**223.3**	**223.3**	*223.3	0.0	0	0	**223.3**	*223.3	0.0	120
Solroed4	243	54	296	2354.2	736.6	†2565.5	248.3	>3600	2	**2358.2**	*2401.2	1.8	>3600
Solroed5	243	20	297	**2054.9**	1843.1	2187.6	18.7	>3600	578	2052.8	*2096.6	2.1	>3600
Solroed6	215	17	266	**1775.5**	1694.5	1833.5	8.2	>3600	846	1774.5	*1802.7	1.6	>3600
Solroed7	194	17	242	**1679.2**	1534.9	1744.7	13.7	>3600	1120	1676.5	*1703.5	1.6	>3600
Vejen1	41	10	58	**424.2**	**424.2**	*424.2	0.0	1	105	**424.2**	*424.2	0.0	137
Vejen2	126	19	159	**1198.5**	1184.2	1225.0	3.5	>3600	1182	1197.1	*1224.6	2.3	>3600
Vejen3	125	19	155	1204.6	1186.9	1234.8	4.0	>3600	1110	**1207.4**	*1214.3	0.6	>3600
Vejen4	125	19	155	1172.5	1145.2	1206.0	5.3	>3600	285	**1182.5**	*1182.5	0.0	852
Viborg1	105	19	133	1034.2	1011.4	1099.6	8.7	>3600	403	**1034.9**	*1034.9	0.0	1518
Viborg2	187	49	230	**2060.0**	1778.8	†2152.4	21.0	>3600	4	2057.8	*2108.2	2.4	>3600
Viby1	124	20	158	1256.9	1255.0	†1279.1	1.9	>3600	582	**1260.5**	*1266.3	0.5	>3600
Viby2	93	13	111	**957.5**	**957.5**	*957.5	0.0	2	0	**957.5**	*957.5	0.0	122
Viby3	45	8	62	**480.0**	**480.0**	*480.0	0.0	1	14	**480.0**	*480.0	0.0	131
Viby4	93	16	111	1053.3	**1053.5**	†*1053.5	0.0	3	2	**1053.5**	*1053.5	0.0	125
Viby5	123	21	156	1356.1	1355.6	†1374.3	1.4	>3600	554	**1356.7**	*1364.6	0.6	>3600

† A wrong upper bound was reported in Kristiansen et al. (2013).

Table 12.4: Summary of results for SCTP. Columns are equivalent to those in Table 12.2.

	Best obj	Best UB	Gap = 0%	Gap ≤ 2%	Gap ≤ 5%	Avg. Gap to best UB
ALNS	37	-	-	-	-	1.26%
Gurobi	16	14	10	22	42	7.13%
B&P	68	95	23	80	97	1.15%

Also for the SCTP the B&P algorithm performs better than Gurobi, providing the best solution in 68 cases. Also in terms of bounds, the B&P algorithm outperforms Gurobi as it finds the best bound in 95 cases. On average, the B&P algorithm is 1.15% within optimum when comparing with the best found bounds which is slightly better than the ALNS algorithm. This is also significantly lower than the 7.13% obtained by Gurobi.

12.6 Conclusion

In this paper we have presented a generalization of a whole family of planning problems, the Generalized Meeting Planning problem. A Branch & Price algorithm has been presented which were tested on two different versions of the Generalized Meeting Planning problem: The Parental Consultation Timetabling Problem and the Supervisor Consultation Timetabling Problem. For both problems the developed B&P algorithm is tested on 100 real-world data examples from Danish high schools. The B&P algorithm on average obtains a gap of 2.32% for the Parental Consultation Timetabling Problem and 1.15% for the Supervisor Consultation Timetabling Problem. These are convincing results for effectiveness of the algorithm. We find it likely that there are many other problems where the Generalized Meeting Planning problem is applicable, and the described B&P approach therefore can be applied.

Bibliography

N.-C. F. Bagger. Generalized Meeting Planning using Mathematical Programming. Technical report, DTU-Management, 2012.

L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In *Practice and Theory of Automated Timetabling III*, pages 104–117. Springer, 2001.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013.

R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR spectrum*, 30(1):167–190, 2008.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. ISSN 0030364x, 15265463.

A. Mehrotra and M. A. Trick. A branch-and-price approach for graph multi-coloring. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, pages 15–29, 2007.

M. W. Padberg. Perfect zero–one matrices. *Mathematical Programming*, 6(2):180–196, 1974. ISSN 00255610, 14364646.

T. C. Pais and P. Amaral. Managing the tabu list length using a fuzzy inference system: an application to exams timetabling. In *The 7th International Conference for the Practice and Theory of Automated Timetabling*, pages 1–6, 2008.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3):230–238, 2003.

R. Qu, E. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee. A survey of search methodologies and automated approaches for examination timetabling. *Computer Science Technical Report No. NOTTCS-TR-2006-4, UK*, 2006.

A. Qualizza and P. Serafini. A column generation scheme for faculty timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 161–173. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

S. Røpke. An era in vehicle routing research is coming to an end: the full solomon test set is solved to optimality, October 2012. Presentation in Recent Research Results in Operations Research at Department of Manegement Science at the Technical University of Denmark.

S. Røpke. Private communication, April 2013.

D. Ryan and J. Falkner. On the integer properties of scheduling set partitioning models. *European Journal of Operational Research*, pages 442–456, 1988.

D. Ryan and B. A. Foster. Integer programming approach to scheduling. *Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling: Papers Based on Presentations at the International Workshop.*, pages 269–280, 1981.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 1997.

J. Thompson and K. A. Dowsland. General cooling schedules for a simulated annealing based timetabling system. In *Practice and Theory of Automated Timetabling*, pages 345–363. Springer, 1996a.

J. M. Thompson and K. A. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations research*, 63(1):105–128, 1996b.

Part III

Other Contributions

Chapter 13 Paper H

Elective Course Planning

Simon Kristiansen[1,2], Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

Abstract Efficient planning increasingly becomes an indispensable tool for management of both companies and public organizations. This is also the case for high school management in Denmark, because the growing individual freedom of the students to choose courses makes planning much more complex. Due to reforms, elective courses are today an important part of the curriculum, and elective courses are a good way to make high school education more attractive for the students. In this article, the problem of planning the elective courses is modeled using integer programming and three different solution approaches are suggested, including a Branch-and-Price framework using partial Dantzig-Wolfe decomposition. Explicit Constraint Branching is used to enhance the solution process, both on the original IP model and in the Branch-and-Price algorithm. To the best of our knowledge, no exact algorithm for the Elective Course Planning Problem has been described in the literature before. The proposed algorithms are tested on data sets from 98 of the 150 high schools in Denmark. The tests show that for the majority of the problems, the optimal solution can be obtained within the one hour time bound. Furthermore the suggested algorithms achieve better results than the currently applied meta-heuristic.

Management of high schools in Denmark has become increasingly complex during the last decade, due to a number of economical and educational reforms. Furthermore the high schools, 10th to 12th grade, became self-governing institutions as of 1/1 2007. Hence the administration of a high school in Denmark has lately been given much more freedom to manage the high school. On the other hand, the high school management now has a much higher economical responsibility and the high schools can even become insolvent, which has indeed happened in a few cases. The by far most important income for the high schools is the grant received from the government. Due to the recent reforms, high schools receive this grant based on the number of students graduating. This has forced the high school management to focus on attracting students, while reducing teaching and administration costs. In this paper we present the *Elective Course Planning Problem* (ECPP), which is of crucial importance for maintaining student contentment.

Due to the special structure of the Danish high schools, literature concerning the ECPP is very limited. However, problems such as *Course Timetabling* and *Student Sectioning* have been looked in to, see e.g. Laporte and Desroches (1986); Tripathy (1984); Erben and Keppler (1996); Rudova and Murray (2003); Müller and Murray (2010). The problem in Tripathy (1984) is actually closely related to the ECPP of this article, but does however lack many of the relevant constraints.

The *International Timetabling Competition 2007* had two tracks focusing on course timetabling, *Post Enrolment based Course Timetabling* and *Curriculum based Course Timetabling*. The first track considers the problem of construction a timetable according to the choice of lectures of the students (Lewis et al. (2007); Müller (2009)), while the second considers weekly scheduling of lectures within a given number of rooms and time periods (Gaspero et al. (2007); Müller (2009)). Neither of these problems matches the ECPP, which is the problem of fulfilling as many elective course requests as possible. I.e. the ECPP is a matter of deciding which course requests should be fulfilled, while the course timetabling problems of ITC2007 are concerned with constructing a timetabling according to some predefined assignments to courses. Still, the ECPP does in fact share several constraints with the mentioned problems, e.g. students can only attend one course at a time. But as will be described in Section 13.1, the ECPP includes other important constraints which are not part of any of the tracks of ITC2007. The timetabling part of the ECPP lies in the fact that, even though the objective is to maximize the number of fulfilled course requests, the course requests should be able to fit in a number of predefined time-blocks.

Both de Werra (1985) and Costa (1994) deals with the problem of first assigning teachers to classes and then assigning classes to a schedule. However, in the Danish education system, the ECPP must be dealt with before allocating teachers to classes. Müller et al. (2007) combines University Course Timetabling and Student Sectioning in an online system based on an *Iterative Forward Search* algorithm. Neither this combination of the two problems proves very resourceful for the ECPP. There exist more practical based publications on solving problems concerning Course Timetabling and Student Sectioning, but it seems none of these are directly related to the ECPP.

Binzer and Kjeldsen (2008) is by far the most relevant source for this article, since it describes the ECPP in detail, and solves the problem using both *Tabu Search* and *GRASP*. Both these meta-heuristics rely on a complicated neighborhood, based on a number of simple moves. Binzer and Kjeldsen (2008) points to the fact that the ECPP has features usually discarded in timetabling literature. For instance, the ECPP is concerned with both optional assigning to time slots and division of classes. Most literature is only concerned with one of these aspects.

Notice that Binzer and Kjeldsen (2008) applies heuristics. In this paper, exact solution methods will be attempted. To the best of our knowledge, no exact algorithm for the ECPP has been described in the literature before.

There are two reasons why solving the ECPP to optimality is so crucial to the high schools. First of all, each offered elective class costs app. 200,000 DKK (28,500 EUR) per year to complete. These costs are fixed costs to be paid, no matter how many students are actually enrolled in the course. If the school has too many courses running with a low number of students it will hence inflict a heavy financial burden on the school. Secondly, if a student is not granted its elective course requests, the student may decide to switch to another high school, which is highly undesirable for the high school administration. Given these important trade offs it is natural that the high schools devote significant resources to find good elective course plans. This is either done manually (typically requiring 2 weeks of full time work) or using existing software solutions. Today most of the high schools (87%) use the cloud-based MaCom Lectio software system for all administrative purposes, and Lectio includes a solver for the ECPP, which is based on the meta-heuristics from Binzer and Kjeldsen (2008). It should be mentioned that the solver in Lectio is highly effective compared to the approach of solving it manually, however in this paper we will show that better algorithms can be found.

The structure of this article is as follows: The ECPP is described in detail in Section 13.1. In Section 13.2 a MIP model is presented. In Section 13.3 the ECPP MIP model is Dantzig-

Wolfe decomposed and a Branch-and-Price algorithm is designed. Furthermore, Section 13.3 presents an alternative approach, Explicit Constraint Branching. In Section 13.4 the algorithm is compared to an existing meta-heuristic for the ECPP, on real-life data from 98 Danish high schools. Finally a conclusion is attempted in Section 13.5.

13.1 Problem Description

In Denmark there exists several types of upper secondary educations. The focus in this paper is STX (Upper Secondary School Leaving Examination). STX is a broad general education and it is the most common type of secondary education in Denmark. The education takes 3 years and consists of 13 to 16 courses.

Once a student has chosen a type of upper secondary education, he needs to choose a study line which fit his interests. Different high schools offer different study lines, so the students choice of high school is somewhat correlated with the choice of study line. It is important to notice that the students can choose free of charge among all Danish high schools. Each study line consists of both a number of mandatory courses and a number of elective courses. When the student has chosen a study line, he is assigned a so-called common class. A common class always consists of students from the same study line. The idea is that courses should be taught in classes with as many students as possible from the same common class, to facilitate the cooperation and social interaction between students.

It should be noted that some courses have a duration of more than one year. This yields many pre-assignment of students to classes, which from this point on will be known as existing classes.

The elective courses give the students some influence on the contents of their education. An elective course can either be an upgrade of a mandatory course to a higher educational level, or it can be a course with a subject the students has not been taught before. Furthermore the students have the opportunity to select an elective course as their second priority. I.e. if an elective course request is not granted, it is possible to assign the student to his second priority course instead. The priority of elective course requests is however omitted in this article as the test data did not contain information regarding priority. A solution-approach for the prioritized requests would be to weight the requests based on their priority. In this article all requests are equally weighted.

13.1.1 Weekly Schedule

In a typical weekly schedule for a high school, each day consists of four modules where teaching is performed. The ECPP is concerned with assigning so-called blocks to modules in the weekly schedule. Blocks consist of a given number of classes teaching specific courses. The students cannot be taught two courses simultaneously, so they must only be part of one class in each block. Binzer and Kjeldsen (2008) refers to blocks as *finished chunks of a time plan*, which seems as a proper description. Figure 13.1 illustrates how five different blocks are assigned to modules.

The elective courses are placed in special blocks in the weekly schedule, e.g. the grey blocks in Figure 13.1. The mandatory courses will hence be taught in the white blocks in the schedule of Figure 13.1. As teaching of the mandatory courses and elective courses can be said to be independent in the weekly schedule, we will in the remainder of the article ignore the problem of planning the mandatory courses.

	Monday	Tuesday	Wednesday	Thursday	Friday
8.15 9.45	BLOCK1			BLOCK3	
10.00 11.30					BLOCK5
	Lunch break				
12:00 13:30					
13:45 15:15		BLOCK2		BLOCK4	

Figure 13.1: Assigning blocks to modules in a weekly schedule

Ideally all elective course requests should be fulfilled. However this is usually unrealistic, not only because of a number of resource limitations of the school, but also due to regulations enforced by education policies. Some common resources limitations are:

- Availability of classrooms. The number of classrooms at the high school is limited. Furthermore some course subjects may require special classrooms, e.g. physics and music education.
- Availability of teachers. It is obvious that a block cannot contain more physics classes than the number of physics teachers available. Furthermore teachers are limited to a certain number of teaching lessons per week.
- Limitations on class sizes.

To model the resource limitations, the concept of subjects is introduced. Let each course be associated with exactly one subject. Resource limitations can then be modeled by specifying a maximum number of courses of a given subject in a block. It is hence assumed that no resource is shared between subjects. This assumption is not too realistic, however it is necessary to compare our results with the existing meta-heuristic algorithm. In a better modeling approach, constraints for each resource should be part of the model. Although it should be noted that only few of the 98 obtained datasets actually specifies this maximum number of classes for any subject, so the negative impact of this assumption is limited.

13.1.2 The Problem

The objective of the ECPP can be stated as follows: Fulfill as many elective course requests as possible using a minimum number of blocks and a minimum number of classes, and assign all existing classes to a block. This suggests that the problem is multi objective, where each of the following is part of the objective:

- Minimize the number of created classes.
- Minimize the number of blocks used for elective courses.
- Maximize the number of elective course requests granted.

Due to compatibility with Binzer and Kjeldsen (2008), it is chosen to keep both the number of created classes and number of blocks constant. Hence the objective is to maximize the number of fulfilled student requests, given a choice of number of classes and blocks. A subject for future research would be to model this with multiple objectives.

Due to resource limitations and limits on the both number of blocks and classes, our mathematical formulation will not guarantee that students are granted the required courses. Consider the following case. A majority of the students in their first school year needs to choose either German B or French B. If however only a single student chooses French B, then this student will in practice never be granted his request, as no high schools will create a class with just one student. The high school has a number of options in this scenario:

- Convince the student to select another course, e.g. German B, instead.
- Convince the student to select a different study line where French B is mandatory.
- Assign the student to a class teaching French B, but where all the other students are from a different common class. Possibly even violate the upper limit on class size to fit the student in.
- The student chooses another high school, which is able to fulfill his requests.

All of these options have a downside which we will not discuss. The point is that these special scenarios cannot be taken into account by a mathematical model, as the possibilities are very diverse and wide-ranging.

13.2 Modeling of Elective Course Planning

The ECPP is now formulated as an IP model which aims at maximizing the number of granted course requests while respecting the conditions. For a particular high school there is a set of students $s \in S$, a set of courses offered $c \in C$ and a set of blocks $b \in B$. Each course belongs to one of the subjects $f \in F$. The maximum number of classes of each subject f in a block is given by $SM_f \in \mathbb{Z}^+$. The relationship between course c and subject f is defined by $SC_{c,f}$. Each student chooses a set of elective courses which he or she wishes to follow, given by $V_c^s \in \{0,1\}$ which takes value 1 if student s has chosen course c, and zero otherwise. For each class there is a lower bound and an upper bound on the number of students, $L_c \in \mathbb{Z}$ and $U_c \in \mathbb{Z}$ respectively. Naturally it applies that $L_c \leq U_c$. The number of classes which can be established is given by the number Q. Finally, there is a set of existing classes $t \in T$. Each existing class contain a set of students $E^{s,t} \in \{0,1\}$, which takes value 1 if student s is part of existing class t, and 0 otherwise. $H_c^t \in \{0,1\}$ is 1 if existing class t is teaching course c. The decision whether a student s should be assigned to course c in a block b, is defined by the binary variable $x_{c,b}^s$. The number of necessary elective course classes to form in block b of course c is given by the integer variable $y_{c,b} \in Z^+$. Finally, the variable u_b^t takes value 1 if existing class t is placed in block b, and 0 otherwise.

The entire MIP model for the ECPP is given in model (13.1).

IP Model for the ECPP (13.1)

$$\max \quad \sum_{c,b,s} x^s_{c,b} \tag{13.1a}$$

$$\text{s.t.} \quad \sum_{c} x^s_{c,b} + \sum_{t} E^{s,t} \cdot u^t_b \leq 1 \qquad \forall\, b, s \tag{13.1b}$$

$$\sum_{b} x^s_{c,b} \leq V^s_c \qquad \forall\, c, s \tag{13.1c}$$

$$\sum_{s} x^s_{c,b} \geq L_c \cdot y_{c,b} \qquad \forall\, c, b \tag{13.1d}$$

$$\sum_{s} x^s_{c,b} \leq U_c \cdot y_{c,b} \qquad \forall\, c, b \tag{13.1e}$$

$$\sum_{c,b} y_{c,b} \leq Q \tag{13.1f}$$

$$\sum_{c} SC_{c,f} \cdot y_{c,b} + \sum_{c,t} SC_{c,f} \cdot H^t_c \cdot u^t_b \leq SM_f \qquad \forall\, b, f, SM_f > 0 \tag{13.1g}$$

$$\sum_{b} u^t_b = 1 \qquad \forall\, t \tag{13.1h}$$

$$x^s_{c,b} \in \{0,1\} \tag{13.1i}$$

$$y_{c,b} \in \mathbb{N}_0 \tag{13.1j}$$

$$u^t_b \in \{0,1\} \tag{13.1k}$$

The objective simply sums up the number of student requests it was possible to satisfy. The constraint (13.1b) ensures that no student is taught simultaneously in two classes, no matter if the classes are elective classes or existing classes. Constraint (13.1c) ensures that a student is only granted an elective course if requested, and each request is only granted once. The constraints (13.1d) and (13.1e) sets the lower and upper bound respectively on the number of students in elective course classes. Constraint (13.1f) limits the maximal number of elective course classes which can be offered. Constraint (13.1g) ensures that the resource limit on subject f is respected. Finally constraint (13.1h) ensures that all existing classes are placed in a block.

The ECPP has been proven $\mathcal{NP}$-hard with a $\mathcal{NP}$-complete decision problem in both Binzer and Kjeldsen (2008) and, in a slightly varied form, in Kristiansen and Sørensen (2010), and the problem contains more than 150.000 binary variables in several of the instances tested. Furthermore the model also contains a great deal of symmetry, since blocks are interchangeable and students who request identical courses are also interchangeable. Notice that all requests provide the same contribution to the objective when granted. See Kaibel et al. (2007) and Margot (2003) regarding solving large integer linear programs with much symmetry. We can not expect that we are able to solve the model with a standard IP solver for problem of non-trivial size.

13.3 Solution algorithms

In this section we will describe a Dantzig-Wolfe decomposition of the ECPP and an alternative approach where Explicit Constraint Branching is applied. Dantzig-Wolfe decomposition is a well-known approach applied to hard optimization problems (see e.g. Vanderbeck (2000)). For an introduction to Dantzig-Wolfe decomposition we refer to Desrosiers and Lübbecke (2005); Dantzig and Wolfe (1960). The ECPP can be Dantzig-Wolfe decomposed in a number of different ways. Decomposition by classes has been found to be most intuitive, i.e. a subproblem is formed

for each combinations of a course and a block. This decomposition is done partially, such that the part of the model concerning existing classes is left as-is. A solution to a subproblem is hence a set of students attending the given course in the given block. It will be shown that this decomposition approach leads to subproblems which we are able to solve to optimality with a greedy algorithm.

13.3.1 The Master Problem

It is the job of the master problem to select those columns, generated by the subproblems, which fulfills as many elective course requests as possible, while maintaining the necessary constraints. Each column corresponds to a *number* of classes of a specific course given in a specific block. To enumerate the columns, a new index for each course block pair (c, b), $p \in P_{c,b}$ is used. For each column a binary variable $z^p_{c,b} \in \{0, 1\}$ is representing the usage of the corresponding column. If a student s is assigned to the course c in block b in column p the constant $A^{s,p}_{c,b} \in \{0, 1\}$ is equal to 1 and otherwise zero. The number of classes which is required for the course c in block b in column p is given by the constant $D^p_{c,b} \in Z^+$. With these definitions, the full master model (13.2) is created.

Decomposed ECPP - Master Problem (13.2)

$$\max \sum_{c,b,s,p} A^{s,p}_{c,b} \cdot z^p_{c,b} \tag{13.2a}$$

$$\text{s.t.} \sum_{c,p} A^{s,p}_{c,b} \cdot z^p_{c,b} + \sum_{t} E^{s,t} \cdot u^t_b \leq 1 \quad \forall\, b, s \tag{13.2b}$$

$$\sum_{b,p} A^{s,p}_{c,b} \cdot z^p_{c,b} \leq V^s_c \quad \forall\, c, s \tag{13.2c}$$

$$\sum_{c,b,p} D^p_{c,b} \cdot z^p_{c,b} \leq Q \tag{13.2d}$$

$$\sum_{c,p} SC_{c,f} \cdot D^p_{c,b} \cdot z^p_{c,b} + \sum_{c,t} SC_{c,f} \cdot H^t_c \cdot u^t_b \leq SM_f \quad \forall\, b, f, SM_f > 0 \tag{13.2e}$$

$$\sum_{b} u^t_b = 1 \quad \forall\, t \tag{13.2f}$$

$$\sum_{p} z^p_{c,b} \leq 1 \quad \forall\, c, b \tag{13.2g}$$

$$z^p_{c,b} \in \{0, 1\} \tag{13.2h}$$

$$u^t_b \in \{0, 1\} \tag{13.2i}$$

The objective function of the master problem, like objective function (13.1a), calculates the number of satisfied elective course requests. Constraint (13.2b) ensures that no student is taught in two courses in the same block. Constraint (13.2c) ensures that an elective course request is only satisfied once. Constraint (13.2d) ensures that no more than Q classes are established. Constraint (13.2e) ensures that the resource limitations of subject f are not exceeded. Constraint (13.2f) ensures that the existing classes are taught. Finally Constraint (13.2g) ensures that only one non-zero $z^p_{c,b}$ is applied for each pair (c, b). This is the convexity constraint.

The full master problem (13.2) is analogous to the direct model (13.1). Unfortunately the master problem has an exponential number of variables, hence column generation is applied to solve the relaxed master problem where the $z^p_{c,b}$ variables and the u^t_b variables are relaxed such that $z^p_{c,b} \in [0, 1]$ and $u^t_b \in [0, 1]$. Now it is possible to start with a restricted master problem where only a subset of $z^p_{c,b}$ variables are used and more variables are only added if necessary.

The subproblem is described in Section 13.3.2, and is is dependent on the dual variables of the constraints in model (13.2), see Table 13.1.

Table 13.1: List of dual variables and the corresponding bounds for the master problem

Constraint	Dual variable	Bound
(13.2b)	α_b^s	≥ 0
(13.2c)	β_c^s	≥ 0
(13.2d)	γ	≥ 0
(13.2e)	$\delta_{b,f}$	≥ 0
(13.2f)	η^t	free
(13.2g)	$\phi_{c,b}$	≥ 0

13.3.2 Subproblem

A subproblem is defined for each combination of a block and a course, i.e. for each (c, b). Given a course c and a block b the subproblem should find the set of students $S' \in S$ who is taught the course c in block b with the maximal reduced profits. Notice that several classes of course c may have to be given to teach the S' students. Given a fixed block b and a fixed course c the binary variable x^s defines whether student s should be included into the set of students S'. The number of classes which are required to teach the S' students is defined by the integer variable y. With these definitions we can now present the entire subproblem in model (13.3).

Decomposed ECPP - Subproblem (13.3)

$$\max \quad C_{c,b}^{\text{red}} = -\sum_s (\alpha^s + \beta^s - W^s) \cdot x^s - (\gamma + \sum_f \delta_f) \cdot y - \phi \tag{13.3a}$$

$$\text{s.t.} \quad x^s \leq V^s \quad \forall\ s \tag{13.3b}$$

$$\sum_s x^s \geq L \cdot y \tag{13.3c}$$

$$\sum_s x^s \leq U \cdot y \tag{13.3d}$$

$$y \leq Q \tag{13.3e}$$

$$SC_f \cdot y \leq SM_f \quad \forall\ f, SM_f > 0 \tag{13.3f}$$

$$x^s \in \{0, 1\} \tag{13.3g}$$

$$y \in \mathbb{N}_0 \tag{13.3h}$$

The objective function of the subproblem defines the reduced profit for the subproblem, given the current value of the dual variables α_b^s, β_c^s, γ, $\delta_{b,f}$ and $\phi_{c,b}$ from the master problem. Constraint (13.3b) ensures that only students who request the course c can be enrolled into S'. Constraint (13.3c) ensures that enough students are enrolled to satisfy the minimal class size and constraint (13.3d) ensures that no more students are enrolled than the maximal class size times the number of classes. Constraint (13.3e) ensures that the no more than the maximal number of allowed classes are created. Finally constraint (13.3f) ensures that the resource limitations are not exceeded.

13.3.3 Combinatorial solution of the Subproblem

It will now be shown that subproblem can be solved to optimality using a greedy algorithm with a time complexity of $O(|S|\log(|S|))$. This is possible due to the uniform knapsack structure of constraint (13.3d), i.e. all students take up the same amount of space in each elective course class. Therefore a greedy algorithm can be written by simply processing the students in order of their contribution to the objective. So even though a lot of different subproblems exist, it will be possible to solve these in an small amount of time. Remember that a course c and a block b is given explicitly.

The objective of the subproblem is now divided in two parts. I.e. a part defined for each of the variables x^s and y. Let p^s define the contribution student s makes to the objective if he is included in the solution. It is not feasible to include student s in the solution if the student has not requested course c. This is denoted by a contribution to the objective of $-\infty$.

$$p^s = \begin{cases} -(\alpha^s + \beta^s - W^s) & V_c^s = 1 \\ -\infty & \text{Otherwise} \end{cases} \tag{13.4}$$

Likewise r is the contribution to the objective given by an increase of y by one.

$$r = -\gamma - \sum_f \delta_f \tag{13.5}$$

By these definitions the objective can be written as

$$C_{c,b}^{\text{red}} = \sum_s p^s x^s + r \cdot y - \phi \tag{13.6}$$

Notice that, by Table 13.1, $r \leq 0$, whereas p^s has unrestricted sign. This entails the following observation: The objective of the subproblem is to find those students which maximizes $\sum_s p^s$, while y should be selected as low as possible, i.e. the students should be fitted into as large classes as possible. The maximal number of classes N which can be created for a course c and a block b is given by equation (13.7).

$$N = \min\left(Q, SM_{\bar{f}}\right) \tag{13.7}$$

Given these definitions we are now ready to present a combinatorial algorithm for finding the optimal set of students S^{opt} which corresponds to the optimal solution of the subproblem, see Algorithm 1. The algorithm gradually builds up a set of students S^{opt} which constitutes an optimal solution. First all the students with positive contributions p^s are included, line 5-7. If the number of student in the class S' is below the lower class limit, extra students are added in line 8-13. Finally it is checked if the group of students in S' will improve the current solution S^{opt}, if yes S' is included into the solution otherwise the algorithm terminates.

13.3.4 Solving the ECPP by Column Generation

The Dantzig-Wolfe decomposition of the ECPP, defined by the relaxed master problem from model (13.2) and the subproblem model (13.3), is solved using the standard Column Generation framework (see e.g. Barnhart et al. (1998)). A simple heuristic is used to find the initial feasible solution. For each course/block pair, create a column which contains all the students which have elected this course. As all constraints of the decomposed part of the problem are set packing constraints, no columns are needed to ensure feasibility.

Algorithm 1 Revised Subproblem (c, b)

input: p^s, r
output: Optimal set S'
Let $\bar{s}$ denotes the student with highest p^s value, not already included in $S^{opt} \cup S'$
$y = 0, S^{opt} = \{\}$
while $y < N$ **do**
    $S' = \{\}$
    while $|S'| < U$ **and** $p(\bar{s}) > 0$ **do**
        Add $\bar{s}$ to S'
        Update $\bar{s}$
    end while
    if $|S'| < L$ **then**
        Calculate $n = |S^{opt}| + |S'| - L \cdot (y + 1)$
        Let $\hat{s}$ denotes the student with highest p^s value, not included in $S^{opt} \cup S'$
        while $n < 0$ **do**
            Add $\hat{s}$ to S'
            Update n and $\hat{s}$
        end while
    end if
    if $\sum_{s \in S'} p^s + r > 0$ **then**
        $S^{opt} = S^{opt} \cup S'$
        $y = y + 1$
    else
        STOP
    end if
end while
if $\sum_{s \in S'} p^s + r - \phi > 0$ **then**
    Add column of S^{opt} to master problem
end if

13.3.5 Explicit Constraint Branching

Explicit Constraint Branching (ECB) is a rather new technique used to improve the performance of Branch and Bound algorithms. The idea is to divide the set of variables into smaller subsets and explicitly add constraints to the problem, and possible also new variables. These additions equip the model with additional structure which may provide superior branching. The approach is generally used when the problem structure required for general constraint branching is lacking. ECB is originally developed for standard MIP solving, see Appleget and Wood (2000).

Suppose a generic MIP is solved, with the integer variable $x_j \geq 0$, $j \in J$. Then let $J' \subseteq J$ be a subset of J and define integer coefficient α_j for each $j \in J'$ such that $\sum_j \alpha_j x_j$ must be integer in any solution to the MIP. A branching scheme to this problem is derived from

$$\sum_{j \in J'} \alpha_j x_j \leq m \qquad \text{or} \qquad \sum_{j \in J'} \alpha_j x_j \geq m + 1 \tag{13.8}$$

where m is any integer. This implies the following steps, which constitute ECB.

- Define the subsets of J_k of the index set J of integer variables.
- Define the integer coefficients $\alpha_{k,j}$ for each k and for each $j \in J_k$.
- Define the general integer ECB variables y_k for each k.
- Add the ECB constraints.

$$\sum_{j \in J_k} \alpha_{k,j} x_j - y_k = 0 \qquad \forall\, k \tag{13.9}$$

Constraint branching is then performed by standard variable branching on the variable y_k. It should be mentioned that the subsets J_k can even be defined dynamically such that these are updated in each iteration of the branching procedure. The idea is that the variables should be roughly evenly divided such that in each subset there is equally many fractional variables, variables with value zero and variables with value one. Moreover if none of $\sum_{j \in J_k} x_j$ is fractional then one or some of the fractional variables should be moved between the subsets such that every sum is fractional.

Keeping the subsets J_k static throughout the process is known as static ECB. The following is the basic ECB constraint,

$$\sum_{j \in J} x_j - y_j = 0 \tag{13.10}$$

which is simply defined over the entire set of variables J. In principle the ECB constraints can be defined over the already existing indices in the model. This approach is applied to the basic ECPP model (13.1), by the following steps:

- Define static ECB-constraint by

$$\sum_{b} y_{c,b} - o_c = 0 \qquad \forall c \tag{13.11}$$

  where the ECB variable to be branched on is $o_c \in \mathbb{N}_0$. I.e. o_c is the number of classes teaching course c. According to Appleget and Wood (2000) branching over this kind of variable breaks some of the symmetry of the model.

- Solve the model with a standard MIP solver with branching priority for o_c set to highest among all variables.

13.3.5.1 Using ECB in a Branch and Price Context

ECB can indeed also be used in a Branch and Price framework, where branching is performed on the master problem. Again we choose to apply constraints which are defined over the already existing indices. For instance the following,

$$\underline{\rho}_c \leq \sum_{b,p} D^p_{c,b} z^p_{c,b} \leq \overline{\rho}_c, \qquad \forall c \tag{13.12}$$

which defines a lower and an upper bound on the total number of classes teaching course c. Initially $\underline{\rho}_c = 0$ and $\overline{\rho}_c = Q$ and branching is performed by changing these bounds. Suppose that for a given c expression (13.12) takes the fractional value l. The tree is now split by the two conditions

$$\underline{\rho}_c = \lceil l \rceil \quad \text{and} \quad \overline{\rho}_c = Q \tag{13.13}$$

$$\underline{\rho}_c = 0 \quad \text{and} \quad \overline{\rho}_c = \lfloor l \rfloor \tag{13.14}$$

which excludes the fractional solution from the feasible area, but maintains all integer solution. Notice that condition (13.13) will potentially result in an infeasible MP. Therefore dummy columns should be added such that feasibility is maintained. Another constraint which could be used exactly the same way is the following

$$\sum_{c,p} D^p_{c,b} z^p_{c,b} - r_b = 0, \qquad \forall b \tag{13.15}$$

which is the total number of classes being taught in each block. Yet another is

$$\sum_{b,s,p} A^{p,s}_{c,b} z^p_{c,b} - \varrho_c = 0, \qquad \forall c \tag{13.16}$$

which is the total number of student being taught course c. A Branch and Price algorithm using ECB will be tested in Section 13.4. Note that even if all ECB variables are integer it is not guaranteed that the solution is in fact integral. Therefore a Branch and Price algorithm using this form of ECB should also implement a second priority branching scheme, such as standard variable branching. Note that imposing ECB constraints results in very little modification of the DWD. The additional constraints are imposed on the MP, which results in new dual variables. However it is trivial to incorporate this new dual variable in the revised subproblem. E.g. the dual variable of constraint 13.12 is simply added to equation (13.5).

13.4 Results

In the previous sections we have suggested four different algorithms. In the initial tests the Branch-and-Price algorithm without the ECB constraints had inferior performance and will hence not be further tested. The remaining three optimization algorithms which we will test are:

- Solving the basic model (13.1) using a standard MIP solver.
- Solving the basic model (13.1) with both the basic ECB constraint (13.10) and constraint (13.11), using a standard MIP solver.
- Branch and Price algorithm using equations (13.12) and (13.15) in an ECB branching scheme, as described in Section 13.3.5.1.

The three algorithms are tested on datasets from 98 Danish high schools for the year 2008. The current version of the high school administration system Lectio applies the meta-heuristic developed by Binzer and Kjeldsen (2008). Given that the three suggested algorithms can achieve optimal results, it is now possible to evaluate the efficiency of the meta-heuristic in Lectio. Tests are ran on a notebook equipped with an Intel Core2 T7200 CPU @ 2 GHz, 4 GB of RAM, and running Windows Vista 32bit. This particular CPU has 2 cores, which is irrelevant as no parallelization has been implemented. The BnP algorithms have been implemented in Microsoft Visual C# using .NET framework version 3.5. The direct models have been implemented in GAMS 22.6.149. For all tests CPLEX 10.0 has been used as solver.

The number of elective course requests which can be fulfilled depends upon the choice of number of blocks and the choice of number of classes. These quantities are attempted selected such that they both are somewhat binding. In Binzer and Kjeldsen (2008) an inspection of dual bounds with respect to number of elective classes is performed. The conclusion is that increasing the number of elective course classes allows for granting more course requests, which is somewhat obvious. However if the number of classes is selected very high, the elective course planning reduces into the more simple matter of assigning students to blocks. A similar analysis could be made for the choice of number of blocks. This motivates the following choice of the maximum number of elective course classes available, see equation (13.17).

$$Q = \left\lceil \sum_{c} \left(\frac{\sum_{s} V_c^s}{U_c} \right) \right\rceil \tag{13.17}$$

A choice on the number of blocks should also be made. The reason why $|B|$ is not predefined is because the high schools usually select this quantity using an ad-hoc procedure. The high school administration will usually like to see solutions for several values of $|B|$. In the following we attempt to derive a formula which selects a tight value for $|B|$. A tight value is preferred, as it provides more interesting results. Let k_1 denote the highest number of existing classes which a single student is attending,

$$k_1 = \max_{s} \sum_{t} E^{t,s} \tag{13.18}$$

At least k_1 blocks should always be established to ensure feasibility. Let k_2 denote the rounded average number of courses pr. student,

$$k_2 = \text{Round} \left(\frac{\sum_{c,s,t} (V_c^s + E^{s,t})}{|S|} \right) \tag{13.19}$$

By inspection of the data, selecting the number of blocks equal to k_2 yields an extremely tight problem. Therefore it seems appropriate to always select a slightly higher value than this. We have chosen to select the number of blocks using equation (13.20), such that the number of blocks can never be lower than $k_2 + 1$.

$$|B| = \max(k_1, k_2 + 1) \tag{13.20}$$

13.4.1 Performance

Table 13.2 contains the average running time and the gap for all problems. It should be noted that for those problems not solved within one hour, $3600s$ is used for the calculation of the

average time. The table shows that in average Direct/w ECB performs best and BnP/w ECB performs worst. Furthermore it is seen that Direct/w ECB in no cases provided a gap worse than 5%, which is quite low. The BnP/w ECB generally performs bad, and even worse than the pure direct model, which is disappointing.

Table 13.2: Running time and gaps for the algorithms

	Direct	Direct/w ECB	BnP/w ECB
Average time	$1373s$	$1121s$	$1523s$
Average gap	0.9%	0.6%	2.8%*
Max gap	7.0%	5.0%	29.0%

* Three problems did not result in a gap within one hour

Table 13.3 contains the percentage of problems solved to optimality within different time spans. The number of solved problems are indicated in brackets.

Table 13.3: Percentage of solved problems

Time(s)	Direct	Direct /w ECB	BnP /w ECB
< 60	45.9 % (45)	50.0 % (49)	35.7 % (35)
60 - 600	12.2 % (12)	12.2 % (12)	19.4 % (19)
600 - 1800	6.1 % (6)	8.2 % (8)	3.1 % (4)
1800 - 3600	1.0 % (1)	3.1 % (3)	1.0 % (1)
> 3600*	34.7 % (34)	26.5 % (26)	31.6 % (39)

* Problems not solved to optimality within one hour

	No.of student	No.of requests	No.of courses	No. blocks	Objective	Assigned classes	Assigned requests
Vejen	382	586	29	3	69572	36	586
Silkeborg	927	1789	65	5	208203	77	1786
Falkoner	421	1080	49	4	127690	66	1080
Vordingborg	415	1462	61	5	182790	68	1462
Alssund	385	650	31	5	64271	34	645
Holstebro	345	567	18	5	56700	29	567
Frederikssund	159	273	18	4	28690	18	273

Table 13.4: Results for a given set of real-life problems at Danish high schools.

The table also shows that Direct/w ECB clearly performs best, solving 50% of the data instances in less than a minute.

Table 13.5 shows a comparison between the meta-heuristic which is currently applied to the ECPP and the direct method with ECB from this paper. It is seen that the exact method does provide a improvement. Even though the average difference is small it must be considered interesting for the high schools to get access to a better approach. For some schools an improvement of 10 classes could be found, and for these high schools the accessibility for a better solution must be considered very important. It should be noted that the running time for the Meta-heuristic is roughly 2 minutes whereas the running time for the Direct method with ECB is one hour.

Table 13.5: Comparison between Direct /w ECB and the Meta-heuristic solver.

	Direct /w ECB	Meta-heuristic	Abs. diff	Rel. diff
Average	27	28	1.1	2.9

However in empirical experiments performed in Kristiansen and Sørensen (2010) it is shown that for a small sample of datasets, Direct /w ECB also performs better with a running time of 2 minutes. In the majority of cases, a running time of 2 minutes actually provided the same best solution as with a running time of 1 hour.

13.4.2 Extension

It is preferable if elective courses classes consist of students from the same common classes, as this entails several benefits from a social point of view. This is done by extending the ECPP such that it minimizes the total number of represented common classes while simultaneously maximizing the number of granted elective course requests. The extension is added using the ϵ-constraint method (Ehrgott (2000)), such that the problem is multi-objective. Given the result of the Direct/w ECB algorithm, an extended model is formulated, using a new binary variable $v^k_{c,b} \in \{0,1\}$. This variable states whether any student from common class k is taught course c in block b. The objective for the new model is then simply to minimize the total sum of common classes, see equation (13.21).

$$\min \sum_c \sum_b \sum_k v^k_{c,b} \tag{13.21}$$

Instead of the old objective function (13.1a), a new constraint on the number of granted elective class requests is included, see equation (13.22).

$$\sum_c \sum_b \sum_s W^s_c \cdot x^s_{c,b} \geq \epsilon \tag{13.22}$$

Finally a link between the $v^k_{c,b}$ variables and the $x^s_{c,b}$ variables is given, see equation (13.23). The matrix $G^{s,k}$ is an incidence matrix which is 1 if student s is part of common class k and 0 otherwise. The following constraint ensures that if one common class is represented in a course in a block, it is counted.

$$v^k_{c,b} \geq x^s_{c,b} \quad \forall\, c, b, s, k, G^{s,k} = 1 \tag{13.23}$$

The parameter ϵ defines the acceptable number of granted elective courses, compared to the optimal solution of direct model, see equation (13.24).

$$\epsilon = R \cdot \sum_{c,b,s} \bar{x}^s_{c,b} \tag{13.24}$$

By adjusting the ratio R to different values we generate the results in Table 13.6. The closer to 1 the diverge percentage R is the less the extended solution differs from the original solution. The table lists the objective solutions. The total number of represented common classes is shown in brackets. Basic denotes the solution from the model without use of the extension.
It is clearly seen that if the ratio R is low, fewer elective course requests are granted and less common classes are represented. The most interesting result from this test are the difference

Table 13.6: Performance test of common class extension

		Objective				
ID	Basic	$R = 1.00$	$R = 0.95$	$R = 0.90$	$R = 0.75$	$R = 0.50$
Avedøre	347 (54)	347 (39)	330 (32)	313 (29)	263 (21)	174 (12)
Esbjerg G.	417 (85)	417 (46)	397 (36)*	379 (33)*	313 (24)	209 (14)
Metropolitan	419 (65)	419 (52)*	399 (42)*	378 (37)*	315 (28)	213 (16)
Nordsjællands	155 (24)	155 (20)	148 (16)	140 (15)	117 (11)	78 (6)
Nærum	972 (190)	972 (115)	924 (89)	875 (76)	729 (52)	486 (29)
Rysensteen	313 (31)	313 (21)*	298 (19)*	282 (17)	237 (14)	159 (8)
Silkeborg	669 (195)	669 (111)*	636 (85)*	604 (72)	502 (46)	339 (22)
Skanderborg	384 (47)	384 (39)	365 (24)	346 (20)	288 (13)	201 (8)
Stenhus	1056(229)	1056 (144)*	1004 (109)*	951 (92)	792 (63)	528 (33)
Støvring	334 (58)	334 (36)	318 (31)	301 (27)	251 (19)	167 (11)
Aalborg	880 (182)	880 (139)*	836 (103)*	792 (88)	660 (60)	440 (33)
Aalborghus	370 (75)	370 (49)	352 (38)	334 (33)	280 (24)	189 (13)
Average	526 (103)	526 (68)	501 (52)	475 (45)	396 (31)	265 (17)

* Not solved to optimality within one hour

between Basic and $R = 1.0$. The number of granted elective course requests is the same, but there is a big difference in the number of represented common classes. On average 103 common classes are represented using the basic ECPP whereas with a diverge percentage of 1 only 68 common classes are used. An improvement of 34%. For some of the larger data sets the improvement are more than 40%. The reason that the extended model is able to improve the solution so significantly is due to symmetry. As mentioned previously the ECPP contains a great deal of symmetry. The extension takes advantage of this to swap students such that the number of represented common classes are minimized while the number of granted requests remains high.

13.5 Conclusion

In this article we have demonstrated how a critical planning problem for the Danish high schools can be optimized by applying three different approaches: Direct MIP model solution, MIP model solution with Explicit Constraint Branching and Branch-and-Price with Explicit Constraint Branching. The algorithms are tested on 98 data sets from different high schools of school-year 2008. The tests show that Explicit Constraint Branching is an interesting tool. Furthermore, the tests reveal that the current meta-heuristic algorithm can be significantly improved. Furthermore, we have shown that an important secondary objective, minimization of the number of common classes, can improve the solution substantially using the ϵ-constraint method. Finally it should be mentioned that currently the approaches in this article are not implemented in any software tools accessible for the Danish high schools. However, showing that the elective course planning can be improved has indeed raised interest among several high schools.

Bibliography

J. Appleget and R. Wood. *Explicit-Constraint Branching for Solving Mixed-Integer Programs*, chapter 14, pages 245–262. Springer Netherlands, 2000.

C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, March 1998. ISSN 0030-364X.

S. P. Binzer and S. H. Kjeldsen. Metaheuristics for high school planning. Master's thesis, IMM, DTU, 2008.

D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76(1):98 – 110, 1994. ISSN 0377-2217.

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

D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2): 151 – 162, 1985. ISSN 0377-2217.

J. Desrosiers and M. Lübbecke. Selected topics in column generation. *G–2002–64*, 34:1–34, 2005.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2000.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

V. Kaibel, M. Peinhardt, and M. Pfetsch. Orbitopal fixing. In M. Fischetti and D. Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin / Heidelberg, 2007.

S. Kristiansen and M. Sørensen. The class packing problem. Master's thesis, DTU-Management, 2010.

G. Laporte and S. Desroches. The problem of assigning students to course sections in a large engineering school. *Computers & Operations Research*, 13(4):387 – 394, 1986. ISSN 0305-0548.

R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Accounting and Finance Working Papers A2007/3, Cardiff University, Cardiff Business School, Accounting and Finance Section, 2007.

F. Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming*, 98:3–21, 2003. ISSN 0025-5610.

T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172: 429–446, 2009. ISSN 0254-5330.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

T. Müller, K. Murray, and S. Schluttenhofer. University course timetabling & student sectioning system, 2007. Space Management and Academic Scheduling, Purdue University.

H. Rudova and K. Murray. University course timetabling with soft constraints. In *Practice And Theory of Automated Timetabling IV.*, pages 310–328, 2003.

A. Tripathy. School timetabling–a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, 1984.

F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48:111–128, January 2000. ISSN 0030-364X.

Chapter 14 Paper I

International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm

Matias Sørensen[1,2], Simon Kristiansen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

14.1 Introduction

An algorithm based on Adaptive Large Neighborhood Search (ALNS) for solving the generalized High School Timetabling problem in XHSTT-format (Post et al. (2012a)) is presented. This algorithm was among the finalists of round 2 of the International Timetabling Competition 2011 (ITC2011). For problem description and results we refer to Post et al. (2012b).

14.2 Adaptive Large Neighborhood Search

Adaptive Large Neighborhood Search was first developed as a metaheuristic for the class of Vehicle Routing Problems (Pisinger and Ropke (2005); Ropke and Pisinger (2006)). It has been applied for few other problem classes as well, including Project Scheduling (Muller (2009, 2010)), Lot-sizing (Muller et al. (2011)), Optimal Statistic Median Problem (Katterbauer et al. (2012)).

Recently we have developed a framework based on ALNS for solving combinatorial optimization problems (written in C# 4.0). This framework is part of the commercial product Lectio[1], where it is used to solve various practical timetabling problems, see Kristiansen et al. (2011); Sørensen and Stidsen (2012) and Kristiansen and Stidsen (2012).

The pseudo code for a general ALNS algorithm is given in Algorithm 1.

[1] http://www.lectio.dk
Cloud-based administration system for high schools. Developed by MaCom A/S, Vesterbrogade 48 1., 1620 Copenhagen V, Denmark

Algorithm 1 Adaptive Large Neighborhood Search

candidate solution x, remove-methods Ω^-, insert-methods Ω^+
$x_{\text{best}} = x$
while stop-criterion not met **do**
    $x' = x$
    `RemoveStrategy`: select q as some quantity to be removed
    `AdaptiveStrategy`: select remove-method $r \in \Omega^-$ and insert-method $i \in \Omega^+$
    remove requests from x' using $r(q)$
    insert requests into x' using i
    `AdaptiveStrategy`: update performance indicators
    if $c(x') \leq c(x_{\text{best}})$ **then**
        $x_{\text{best}} = x'$
    end if
    `AcceptStrategy`: set candidate solution x to either x', x_{best} or x itself
end while
return x_{best}

The main points of the algorithm are described below in general terms.

- In each iteration, a remove and insertion method is chosen and applied to the candidate solution. The combination of these methods defines the neighborhood of the algorithm, hence there exists $|\Omega^-| \cdot |\Omega^+|$ different neighborhoods.
- `RemoveStrategy`: Governs the selection of q. This has major influence on how much computational time each iteration requires.
- `AdaptiveStrategy`: Responsible for selecting remove and insertion methods in each iteration, and updating their respective performance indicators of these method by some metric.
- `AcceptStrategy`: Determines which solution to use as candidate solution for next iteration. This could in principle be any known solution, but is usually selected as either the current candidate solution x itself, the newly produced solution x', or the current best solution x_{best}.

14.3 Algorithm setup for ITC2011

Here we describe our implementation of a ALNS algorithm for the XHSTT format. The choice of ALNS strategies are briefly mentioned below. More details will be available in the full paper.

- `RemoveStrategy`: The remove and insertion methods deal with sub-events. q is defined as the sum of the duration of the sub-events which are removed from the solution. We select q as a random number, bounded by a percentage of the total duration of all instance events.
- `AdaptiveStrategy`: We have chosen a metric essentially based on two parameters for each method; The number of times the method was part of an iteration which yielded a better solution than the current one, and the relative gap between the current solution and the resulting solution from applying the method.

- `AcceptStrategy`: An acceptance criteria borrowed from Simulated Annealing (SA) is used, with the following additional property: If no new best solution has been found in a number of iterations, the temperature is increased by a factor, and the candidate solution is set to the best known solution. The intention is to allow more diverse exploring of the area around the best known solution, in case the algorithm gets 'stuck'.

Let a *move* be a small perturbation on a solution. The following moves are used in this implementation: Move $M_{se,t}$ denotes the assigning of sub-event se to time t. $M_{r,er,se}$ denotes the assigning of resource r to event resource er on sub-event se. Furthermore we also implement the corresponding unassign-moves, denoted $M^{\neg}_{se,t}$ and $M^{\neg}_{r,er,se}$, respectively.

Using these moves a total of 9 insertion methods (all more or less based on the greedy principle, e.g. regret heuristics (Potvin and Rousseau (1993); Sørensen and Stidsen (2012)), and 14 remove methods (all based on some element of relatedness and an element of randomness) are implemented. These methods are divided into three categories, based on what they (un-)assign: Only times, only resources, or both times and resources.

An example of a remove method is the following, which removes sub-events from non-preferred times: Given an XHSTT instance, and a solution S to this instance. Find all tuples $\langle se, t\rangle$ of S, where sub-event se is assigned time t, and t is not a preferred time for sub-event se (see *Prefer times constraints*, Kingston (2010)). Let the set of these tuples be denoted U. Select randomly a subset of these tuples $\overline{U} \subseteq U$ such that the sum of the duration of all sub-events of the tuples in $\overline{U}$ equals q. Perform an unassign time move $M^{\neg}_{se,t}$ for each of the tuples in $\overline{U}$.

An example of an insertion method is the following: Let $\Delta(M) \in \mathbb{R}$ be the profit of performing move M on the solution at hand S. Select $M_{\text{best}} = \arg\min_{se,t}(\Delta(M_{se,t}))$, and if $\Delta(M_{\text{best}}) \leq 0$, apply M_{best} to S and repeat, otherwise stop. This is a greedy method which assigns times to sub-events, until no profitable move can be found.
In the full paper all insert/remove methods will be described in detail.

The final algorithm contains 9 free parameters, which were tuned for best performance using the `irace` package (see López-Ibáñez et al. (2011); Birattari (2005)).

14.4 Final remarks

This paper documents how Adaptive Large Neighborhood Search can be applied to problems in XHSTT format.

The proposed algorithm was applied to all instances in archive XHSTT-ITC2011, and showed competitive results in most cases (comparing to the best known solutions at that point in time).

ALNS has not been used much in the field of timetabling, but we see no reason to believe that ALNS should not perform well on other (related) problems in this field.

Acknowledgements Thank you goes to Michael Herold for fruitful discussions concerning ALNS strategies. Thank you to Manuel López-Ibáñez for help using the `irace` package. And finally thank you to David Pisinger for advice on ALNS implementation.

Bibliography

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

K. Katterbauer, C. Oguz, and S. Salman. Hybrid adaptive large neighborhood search for the optimal statistic median problem. *Computers & Operations Research*, 39(11):2679 – 2687, 2012. ISSN 0305-0548.

J. H. Kingston. The hseval high school timetable evaluator, 2010. URL `http://it.usyd.edu.au/~jeff/hseval.cgi`.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, Université Libre de Bruxelles, IRIDIA, Av F. D. Roosevelt 50, CP 194/6 1050 Bruxelles, Belgium, Februrary 2011. http://iridia.ulb.ac.be/irace.

L. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

L. Muller. An adaptive large neighborhood search algorithm for the multi-mode resource-constrained project scheduling problem. l, Department of Management Engineering, Technical University of Denmark Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark, 2010.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

M. Sørensen and T. R. Stidsen. High school timetabling: Modeling and solving a large number of cases in denmark. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 359–364. SINTEF, 2012.

Chapter 15 Paper J

Comparing Solution Approaches for a Complete Model of High School Timetabling

Matias Sørensen[1,2], Thomas R. Stidsen[1]
[1]Management Science, Department of Management Engineering,
Technical University of Denmark
[2]MaCom A/S, Vesterbrogade 48 1., DK-1620 Kbh V., Denmark

DTU Technical Report 5.2013 which preceded Paper B. Only parts of the text are replicated here. References to equations in Paper B are used throughout.

15.1 Complexity

Most common variants of non-trivial school timetabling problems have been proved to be $\mathcal{NP}$-hard, see Even et al. (1975); Cooper and Kingston (1996); ten Eikelder and Willemen (2001). However, we have not been able to find a formulation of timetabling which resemble exactly the one of this paper.

To prove that the Danish High School Time Problem (DHSTP) is $\mathcal{NP}$-hard, Proposition 6.3 of Wolsey (1998) is used. I.e. it must be shown that DHSTP is in $\mathcal{NP}$, and that another $\mathcal{NP}$-complete problem can be polynomially reduced to the decision-version of DHSTP (by Definition 6.7 of (Wolsey, 1998, p. 88)).

Let the objective value of model (2.4) be denoted z_{DHSTP}. The decision version of model (2.4) asks whether a solution exists with objective $z_{\text{DHSTP}} \leq k$. Clearly the decision version of DHSTP is in $\mathcal{NP}$ as a solution $x_{e,r,t}$ to model (2.4), can be checked to have objective value less than k in polynomial time.

The well known $\mathcal{NP}$-complete problem Graph k-Colorability problem (GCP) asks whether it is possible to assign each vertex of a graph G a color such that no two adjacent vertices have the same color, using at most k colors. To show that GCP is polynomial reducible to DHSTP, a conversion scheme is now given, which transforms any instance of GCP into an instance of DHSTP. An instance of GCP consists of a graph G with vertices V and edges F, and a number of colors k.

- Start with an empty instance of DHSTP, i.e. $\mathcal{D} = \mathcal{M} = \mathcal{E} = \mathcal{C} = \emptyset$, $\mathcal{T} = \{t_D\}, \mathcal{R} = \{r_D\}$.
- For each vertex $v \in V$, create an event e.

- For each edge $f \in F$ between vertices v_1 and v_2, create an entity a. Let events e_1 and e_2 represent vertices v_1 and v_2, respectively. Assign entity a to both e_1 and e_2, i.e. $B_{e_1,a} = B_{e_2,a} = 1$.
- Create one day d with k timeslots $\{t_0, \dots t_{k-1}\}$.
- Set $\alpha_{e,r,t} = \begin{cases} 0 & t = t_D \\ 1 & \text{else} \end{cases}$, and $\epsilon = \beta_a = \gamma_a = \delta_a = \zeta = \iota = \theta = \eta_a = 0$.
- As the only room in the instance is the dummy-room, the following substitution is made: $y_{e,t} = x_{e,r_D,t}$.

This problem-setting makes a lot of constraints and variables redundant. The DHSTP instance can therefore be written as follows (written as a maximization-problem by changing sign of $\alpha_{e,r,t}$):

DHSTP reduced problem (15.1)

$$\max \quad z_{\text{DHSTPReduced}} = \sum_{e,r,t} \alpha_{e,r,t} x_{e,r,t} \tag{15.1a}$$

s.t.

$$\text{(one time/room)} \sum_{t} x_{e,r_D,t} = 1 \qquad \forall e \tag{15.1b}$$

$$\text{(entity conf.)} \sum_{e} B_{e,a} x_{e,r_D,t} \leq 1 \qquad \forall a, t \neq t_D \tag{15.1c}$$

Solving model (15.1) gives an objective $z_{\text{DHSTPReduced}}$, corresponding to the number of events which are assigned a timeslot different from the dummy-timeslot. By the conversion scheme, an event corresponds to a vertex in graph G and a timeslot corresponds to a color. Therefore $z_{\text{DHSTPReduced}}$ corresponds to the number of vertices which is assigned a color. To answer whether graph G is k-colorable, one can simply check if $z_{\text{DHSTPReduced}} = |V| \leq k$ (corresponding to solving the decision version of model (15.1)).

Any instance of GCP can thereby be converted into an instance of the decision version of DHSTP. Therefore the decision version of DHSTP is $\mathcal{NP}$-complete, so by Definition 6.7 of (Wolsey, 1998, p. 88), DHSTP is $\mathcal{NP}$-hard.

15.2 Conversion to the XHSTT format

The XHSTT format (Post et al. (2012a)) is an XML-based format for (High) School Timetabling problem instances. It was used for the International Timetabling Competition 2011 (Post et al. (2012b)). Currently, 38 non-artificial datasets from 11 different countries are available.

In this section the problem instances of Lectio will be modeled in the XHSTT format. This will allow us to easily publish our instances. However, some aspects of DHSTP cannot currently be modeled with XHSTT, which is discussed in Section 15.2.2. It is assumed throughout this section that the reader has in-depth knowledge of XHSTT.

Times

- `TimeGroups`: One day-`TimeGroup` is created for each day $d \in \mathcal{D}$. Furthermore, if the instance is a two-week instance, a week-`TimeGroup` representing each week is also created.

Furthermore a `TimeGroup` is created for each neighbor-day pair, as described in Section 7.2.2.3, which contain all times of the respective days.

- `Time`: One `Time` is created for each timeslot $t \in \mathcal{T}$.

Resources

- `ResourceTypes`: Three `ResourceTypes` exist, namely `Room`, `Student`, `Teacher`. These correspond analogously to the sets $\mathcal{R}$ and $\mathcal{A}$ (students and teachers).
- `Resources`: For each room $r \in \mathcal{R}$ and each entity $a \in \mathcal{A}$, a `Resource` of corresponding type is created.

Events

- `EventGroups`: For each class $c \in \mathcal{C}$, a corresponding `EventGroup` is created. The members of an `EventGroup` are the events where the class participates. Notice that this is very similar to the definition of `Courses`, but since an event can contain more than one class, we use `EventGroups` instead of `Courses`.
- `Events`: A conversion from DHSTP EventChains to XHSTT-`Events` is now described. Events are either combined into the same event or linked together using constraints (i.e. certain events should be placed in the same or immediately following timeslot as other events).
  - Denote the set of entities, the set of classes, and the set of eligible rooms for event e by $\mathcal{A}_e$, $\mathcal{C}_e$ and $\mathcal{R}_e$, respectively. If for EventChain ec there exists two events $e_1, e_2 \in ec$ for which the set of entities, classes, eligible rooms, or locked room are different, i.e. if $\mathcal{A}_{e_1} \neq \mathcal{A}_{e_2}$, $\mathcal{C}_{e_1} \neq \mathcal{C}_{e_2}$, $\mathcal{R}_{e_1} \neq \mathcal{R}_{e_2}$, or $LR_{e_1,r} \neq LR_{e_2,r}$, then all events in EventChain ec must be linked using constraints.
  - If any event $e \in ec$ should be placed alongside other events, i.e. $S_e \neq \emptyset$, then all events in EventChain ec must be linked using constraints.
  - If none of the above applies, events are combined into one `Event`.

  Figure 15.1 illustrates conversion of some EventChains to XHSTT `Events`.

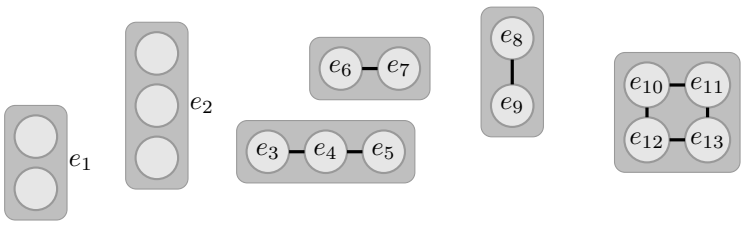


Figure 15.1: Conversion from EventChains to XHSTT events. e_1 and e_2 represent events which are combined into one XHSTT-`Event`. The remaining events are linked together using constraints.

15.2.1 Constraints

In the following constraints of DHSTP is mapped to the XHSTT format. As in the MIP model (2.4), all constraints have `CostFunction` = `Sum`.

15.2.1.1 One timeslot - AssignTime

Only a single `AssignTime` constraint is needed, which applies to all events. `Weight` is set equal to 1, and `Required` equals `true`.

15.2.1.2 One room - AssignResource

A single `AssignResource` constraint is created, which applies to all events, and has `Role` = `Room` and `Required` = `true`. The `Weight` is set equal to 1.

15.2.1.3 Do not split events - SplitEvent

No events should be split, so all events are grouped by their duration, and for each group a single `SplitEvent` constraint is created, which applies to these `Events`, have `Required` = `true`, `Weight` = 1000, `MinimumDuration` and `MaximumDuration` set accordingly, and `MinimumAmount` = `MaximumAmount` = 1.

15.2.1.4 Teacher unavailable times - AvoidUnavailableTimes

The set of unavailable timeslots for a teacher is known (these partly defines parameter $D_{e,t}$). Group teachers by this set of timeslots, and create a `AvoidUnavailableTimes` constraint which applies to these teachers, and the respective set of timeslots. Further, `Required` = `true`, and `Weight` = 1.

Unavailable times for students are skipped as these are usually artificial in the sense that students are only marked as unavailable in certain timeslots by preference. I.e. for students it is preferred that late timeslots on each day are only used if necessary.

15.2.1.5 Do not split EventChains over days - PreferTimes

Neither an event or an EventChain can be assigned timeslots such that it spans over several days. For each event, identify its feasible timeslots by its EventChain. E.g. if an event has events which must be placed in contigious positions, then it cannot be assigned the last timeslot on a day.

Group events by their set of feasible timeslots. Create a `PreferTimes` constraint which apples to the appropriate events and times, has `Required` = `true` and `Weight` = 1.

15.2.1.6 Eligible rooms - PreferResource

Each event must be constrained such that it is only assigned its eligible rooms. Identify a set of `Resources` by parameter $K_{e,r}$, and create an `PreferResource` constraint with `Required` = `true`, `Weight` = 1000 and `Role` = `Room`. If several events have the same set of eligible rooms, these `PreferResource` constraints can be grouped. Notice that the priority of rooms as defined by eq. (7.37) is ignored.

15.2.1.7 Entity and Room conflicts - AvoidClashes

Only one `AvoidClashes` constraint is defined, which applies to all rooms, students and teachers. The constraint has `Required` set to `true` and `Weight` = 1. The XHSTT format does not currently allow us to restrict `AvoidClashes` constraint to only check for clashes in a subset of events, as was done in eq. (7.3′) and eq. (7.4′). Therefore instances might have inevitable violations of hard constraints.

15.2.1.8 Required days off - ClusterBusyTimes

Group teacher-entities by their number of required days off D, skipping those which require no days off. For each of these groups, generate a `ClusterBusyTimes` constraint which applies to these entities, with `Minimum` $= 0$, `Maximum` $= |\mathcal{D}| - D$, `Required` $=$ `true`, `Weight` $= 1$ and `TimeGroups` equal to the set of timegroups representing days.

15.2.1.9 Days occupied penalty - ClusterBusyTimes

Create a `ClusterBusyTimes` constraint which applies to all teacher-entities, with `Minimum` $=$ `Maximum` $= 0$, `Required` $=$ `false`, `Weight` as set by eq. (7.39), and
`TimeGroups` equal to the set of timegroups representing days.

15.2.1.10 Days off penalty - ClusterBusyTimes

Create a `ClusterBusyTimes` constraint which applies to all student-entities, with `Minimum` $=$ `Maximum` $= |\mathcal{D}|$, `Required` $=$ `false`, `Weight` as set by eq. (7.40), and `TimeGroups` equal to the set of timegroups representing days.

15.2.1.11 Neighbor day conflicts - SpreadEvents

Define an `SpreadEvents` constraint which applies to all `EventGroups` representing classes, with `Weight` as set by eg. (7.42) and `Required` $=$ `false`. The `TimeGroups` section contains all `TimeGroups` which define a neighbor-day pair, and all entries have `Minimum` $= 0$, `Maximum` $= 1$. Notice that all neighborday conflicts are penalized for all classes, contrary to eq. (7.29).

15.2.1.12 Penalize idle slots - LimitIdleSlots

Two `LimitIdleSlots` constraints are created, which applies to all student-entities and all teacher-entities, respectively. The `Weight` is set as by eq. (7.38), and both constraints have `Required` $=$ `false`, `Minimum` $=$ `Maximum` $= 0$, and `TimeGroups` representing days.

15.2.1.13 Events in same timeslot - LinkEvents

Events which should be placed in the same timeslot as others can be specified using the `LinkEvent` constraint. For each set of these events, create a `LinkEvents` constraint which applies to these events, with `Required` $=$ `true`, `Weight` $= 1$, and one `EventGroup` which represents all events.

15.2.1.14 Events in contiguous timeslots - OrderEvents

For events which should be placed in contiguous timeslots ('followers'), an `OrderEvents` constraints is created with `Required` $=$ `true` and `Weight` $= 1000$. The constraint applies to all pairs of events (e_1, e_2) where e_2 should follow immediately after e_1. All pairs have `MinSeparation` $=$ `MaxSeparation` $= 0$.

15.2.1.15 Class day conflicts - SpreadEvent

A single `SpreadEvents` constraint is created, which applies to all `EventGroups` representing classes, with `Required` $=$ `true` and `Weight` $= 1$. The `TimeGroups`-section is set equal to the set of timegroups representing days, with every entry having `Minimum` $= 0$ and `Maximum` $= 1$, Notice that it is not possible to constrain the events for which this constraints is applied, as was done in eq. (7.24). This may lead to hard constraint violations if classes are part of several events in

the same EventChain, and these events cannot be combined into the same `Event`, as described in the `Event`-conversion scheme.

15.2.1.16 Daily workload - LimitBusyTimes

Group all teacher-entities by their maximum number of work-hours per day W_a. For each of these groups, create a `LimitBusy` constraint which applies to these teachers, with `Minimum` = 0,`Maximum` = W_a, `Required` = `true`, `Weight` = 1, and timegroups representing days.

15.2.1.17 Week stability - LimitBusyTimes

The week stability constraint (7.35) cannot be modeled entirely as-is. Instead it is assumed that all events of class c is assigned a timeslot. This assumption seems fair in light of the applied AssignTime constraint. Let $d_c = \sum_e J_{e,c}$ be the number of events containing class c. Now group classes by d_c and create a `SpreadEvents` constraint which applies to all `EventGroups` representing these classes, have `Required` = `false` and applies to two timegroups, each representing a week, with `Minimum` = $\lfloor \frac{d_c}{2} \rfloor$ and `Maximum` = $\lceil \frac{d_c}{2} \rceil$. Thereby a class with 5 event must have a week-distribution of 2/3 or 3/2, and a class with 6 events must have the distribution 3/3.

15.2.2 Summary

The following aspect of DHSTP are not modeled with the XHSTT format:

- Penalty for rooms with low priority for events, as defined in eq. (7.37). This is a minor flaw, as few events will have second and third-priority rooms.
- Events can be assigned a room, but not a timeslot. This also clashes with a hard constraint of DHSTP, however, it does not pose a major problem as such events could be filtered out if we imagine solving the XHSTT instance in a practical setting.
- All neighbor-day conflicts are penalized. As the penalty given is small, this is a minor flaw.
- All room conflicts, entity conflicts and class day conflicts are penalized, which might give inevitable violation of hard constraints.
- Constraints room stability ((7.22) and (7.21)) and days off week stability ((7.33)) cannot currently be modeled with XHSTT. Both are small flaws, as these represent soft-constraints with a small weight.
- The combining of students as described in the beginning of Section 7.2.2 is not taken into account. This would be possible by introducing separate constraints for different groups of students.

Even with these inconclusive aspects of XHSTT with respect to DHSTP, we still believe the conversion of Lectio instances have significant contribution. All hard-constraints can be modeled more or less accurate. The soft-constraints which are left out are not of very significant character. Furthermore the resulting datasets are the first ones to use the `OrderEvents` constraint, and the first ones to span multiple weeks.

Bibliography

T. Cooper and J. Kingston. The complexity of timetable construction problems. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 281–295. Springer Berlin / Heidelberg, 1996.

S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:184–193, 1975. ISSN 0272-5428.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

H. ten Eikelder and R. Willemen. Some complexity aspects of secondary school timetabling problems. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 18–27. Springer Berlin / Heidelberg, 2001.

L. Wolsey. *Integer programming*. Wiley-Interscience publication, 1998.