universität wien

MASTERARBEIT

Titel der Masterarbeit

„Air fleet assignment model: an example of a flight school timetabling formulation”

Verfasst von

Maria Golova

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 066 915
Studienrichtung lt. Studienblatt:	Masterstudium Betriebswirtschaft
Betreuer / Betreuerin:	o. Univ.-Prof. Dipl.-Ing. Dr. Richard F. Hartl

ACKNOWLEDGMENTS

First of all, I would like to express my greatest gratitude to o. Univ.-Prof. Dipl.-Ing. Dr. Hartl for his enormous support in writing this thesis and for his creative ideas in solution approaches. I am also thankful to Dr. Gansterer and MSc Schnell for their significant contributions in creating and programming a problem formulation.

Secondly, I would like to thank Mag. Gahbauer for drawing my attention to the existing practical problem and thus for facilitating this thesis topic and research. In addition, I am also grateful for his help in providing thoughtful information about business operations of the flight school.

Finally, I would like to express appreciation to my mother, Mag. L. Aleksejeva, for awakening my aspiration for knowledge from the younger years and constantly fuelling it up throughout my studying time.

Vienna, April 2014

ABSTRACT

This paper creates an integer programming approach to the real-world problem, which partly bases on course timetabling problem, as well as includes unique formulation required by the case. Precisely, the model presented in this paper aims to maximize the amount of monthly held lectures that are assigned according to students' preferences. A presented timetable also depends on a number of hard (e.g. no more than one student in the same plane at the same time) and soft constraints (e.g. no assignment of expensive plane type for the beginner courses) in order to consider diverse restrictions set by a school and hence to design a realistic timetable.

The problem was tested on three different data sets with a help of mixed integer program solver (FICO Xpress). The formulation developed in this thesis has shown good practical results, especially for smaller problems. Medium and large instances were subject to simplification of the formulation due to the need of improvement in computational runtime and solution. Solutions to large problems require further investigation of possible relaxations, as well as the use of non-exact methods in order to improve runtime and method optimality.

DEUTSCHE ZUSAMMENFASSUNG

Diese Arbeit behandelt eine Annäherung an ein reales Problem durch Einsatz von ganzzahliger Programmierung (IP), die zum einen auf das Problem der Kursplanung *(course timetabling)* basiert und zum anderen eine Spezialbehandlung für diesen Fall erfordert. Genauer gesagt, versucht das in dieser Arbeit präsentierte Model die Anzahl der im Monat abgehaltenen Flugstunden zu maximieren. Der vorgestellte Stundenplan hängt von einigen notwendigen Bedingungen (z.B. nicht mehr als ein Student pro Flugzeug zur gleichen Zeit) und hinreichenden Bedingungen (z.B. keine Zuordnung von teuren Flugzeugmodellen für Anfängerkurse) um einerseits Restriktionen der Flugschule zu berücksichtigen und andererseits einen realistischen Stundenplan zu erstellen.

Das Problem wurde anhand von drei verschiedenen Datensätze, mit Hilfe von einem MIP - Solver (FICO Xpress) getestet. Die in dieser Arbeit entwickelte Lösung für die geschilderte Problemstellung zeigt gute Resultate, insbesondere für Probleme kleinerer Art. Mittelschwere und größere Problemfälle wurden in diesem Fall vereinfacht auf Grund der Verbesserung der Lösung und der Berechnungszeit. Lösungen für größere Probleme erfordern weitere Erforschung von möglichen Relaxionen und auch Nutzung von nicht-exakten Methoden um die Berechnungszeit und die Methode zu optimieren.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ATPL	Air Transport Pilot License
FNPT	Flight and Navigation Procedures Trainer
JA	Jet Alliance
ITC	International Timetabling Competition
LP	Linear Programming
MEP	Multi Engine Piston
MIP	Mixed Integer Programming
SEP	Single Engine Piston
SPIC	Student Pilot In Command

1. INTRODUCTION

Air fleet assignment models cover a very broad scope of tasks. The main objective of the most problems is to define an optimal number of flights per day under the consideration of multiple constraints. As a choice of constraints depends solely on the practical requirements of a company with its own regulations, costs, revenues and characteristics, finding a standard formulation that would suit all real-cases is quite impossible. Therefore, according to the actual problem definition the air fleet assignment models may be easily combined with other types of common formulations, in particularly with timetabling problems.

Timetabling problems have been widely studied in the literature during past 50 years (cf. surveys by Schaerf, 1999; Burke et al., 2009; Qu et al., 2009; MirHassani & Habibi, 2013). In general, timetables problems are split into three types: school, course and examination timetabling. In particular, however, no unity among the timetabling researchers exists, as each paper tries to bring up a new formulation that corresponds to each real-case demand. In order to systemize timetabling problems and to create benchmarks for comparison, three International Timetabling Competitions (ITC) were organized (ITC2002, ITC2007, ITC2011). The active participation in the competitions proved once again the breadth and the importance of the timetabling problem family, as well as the need for further research and a search for a common ground.

The aim of this thesis is neither to find a systemization of timetable problems or to achieve the best operational time with the ITC test instances, but rather to create a problem formulation for a real-case that is based on the diverse formulations proposed in the literature. This thesis will try to test whether an individual problem can be adopted to existing formulations and then further expanded with a consideration of unique requirements. Hence, this paper aims finding a gap between theory and practice.

More precisely, the focus of this paper is to create a feasible timetable for a flight school. This timetable should assign students to lectures according to their preferred times, flying skills, as well as corresponding teacher and plane characteristics. The real problem will be studied in detail, so that each practical requirement could be mentioned and formulated. The result of this thesis is a flight

school timetable formulation with a number of hard constraints that are typical for course timetable problems, and soft constraints that are created particularly for this real-case. Afterwards, the problem will be approached with an integer programming and solved with an exact method in a commercial solver.

The research topic of this thesis was motivated by theoretical and practical relevance of the paper induced by an existence of real problem. The interest in creating a timetable for a flight school was enforced during a personal communication with the flight school strategic planning manager, who has enlighted the complexity of manual planning and the necessity for automated planning tools.

This thesis is structured as follows: Chapter 2 gives an overview of research developments in a field of timetabling problems, with a special focus on the course timetable problems. Chapter 3 gives a detailed problem description and discusses practical verbal requirements of the model. Chapter 4 introduces the mathematical model and describes its formulation in detail. Chapter 5 presents real-data sets, as well as computational experiments of the model with provided data. Finally, Chapter 6 concludes the paper.

2. LITERATURE REVIEW

The air fleet assignment models or the aircraft daily routing and scheduling problems deal with many different modelling tasks. The first research in the academic literature on aircraft routing was carried out already in the 1960s. Most papers referring to the earlier research periods describe small airline problems, which aim at determining the number of optimal flights per day on a predetermined route using heuristic solution methods. (Desaulnier et al., 1997) With the development of computers and IT technologies, the air fleet models have also become much more complicated and sophisticated.

The typical fleet assignment problem deals with assigning different aircraft types to the scheduled flights, whereas a diverse number of constraints have to be considered. The constraints most usually include the type of the aircraft, its capacity and characteristics, equipment capabilities and availabilities, operational costs and revenues, time windows and gate preferences, as well as legal air and cross-border regulations. (Sherali et al., 2006)

Another major sub problem in the air fleet modelling field deals with crew assignment to the flight segments. The crew assignment problems help airline companies to allocate their crew most optimally to the different routes, respecting a number of work regulations and collective agreements such as maximal duration of allowed sequential flying hours or the minimum length of rest period between the flights. (Zeghal & Minoux, 2006) Due to the complexity of crew members' characteristics and suitability bounds to different aircrafts, as well as significant number of the regulations rules and working agreements, the crew assignment problems are considered to be difficult to solve. In order to solve such problems, various methods have been suggested in the literature. Researchers (see Desrochers & Soumis, 1989; Vance, et al., 1997; Kohl, 2003) propose to decompose the problem into two sub-problems: the crew pairing problem and working schedule construction problem, which are then solved sequentially. A solution to decomposed problem is then based on the column-generation principle elaborated by Gamache et al. (1994) and further developed into the Branch-and-Bound approach by Stojkovic et al. (1998), Lettovsky et al. (2000), and others.

Zeghal & Minoux (2006) propose a new approach for solving crew assignment problem, in which the problem is not viewed as a crew pairing problem and is capable of solving the two sub-problems simultaneously. In comparison to the previous papers, the authors formulate the problem as a large scale integer linear program combining a number of constraints, which are not solely of the partitioning type as suggested before. Due to a new formulation, the problem with a larger amount of constraints can be solved faster and provide more feasible results.

Nevertheless, the main focus of this thesis is a significantly smaller model, which assigns different aircraft types and pilots to the scheduled flights. The model does not aim to consider a large amount of constraints regarding air fleet regulations, gate preferences as well as revenue and cost information. Therefore, the model can be formulated as a scheduling problem within the aircraft routing model.

The general goal of the scheduling or timetabling problems is to schedule "a sequence of lectures between teachers and students in a prefixed period of time, satisfying a set of constraints of various types" (Schaerf, 1999, p. 1). Starting with Gotlieb (1963), the automated scheduling problems have gained a large attention already in 1960' due to their wide applicability to different tasks. Therefore, the timetabling problems are not only related to lecture or course scheduling, but can be also used for assigning tasks, public events, vehicles or people to a limited number of resources (Lewis, 2007). The timetabling problems arise in a various fields of activities like education, healthcare institutions (e.g. medical staff timetabling), transport (e.g. public transport, train or aircraft timetabling) or sport (e.g. matches between teams or individuals) (Petrovic & Burke, 2004).

Although scheduling and timetabling problems can be regarded as equal terms for a common problem, some researches identify a minor difference between two. According to Wren (1996), the objective function of the scheduling problem is focused on the minimization of the total cost of resources needed, whereas the objective of the timetabling model is to propose a feasible and desirable solution disregarding cost values. Carter (2001) indicates the difference by the way that scheduling problems allocate the resources to a particular task (e.g. particular teacher to a course), while timetabling rather concentrates on the time of the event, where the resources were already set as fix in advance. Nonetheless, the difference between the definition of the scheduling and timetabling is not significant, and thus further in this work, both terms are considered equal.

Further common feature of all scheduling and timetabling problems is a classification into hard and soft constraints. As hard constraints are more essential than soft constraints, the satisfaction of all hard constraints is compulsory for a feasible timetable. (Zhang et al., 2003) Examples of hard constraints may include assigning teacher only to one class (students, plane etc.) at a certain time period, scheduling students to only one exam at a time period or making the room available only to one student group a time period. On the other side, the soft constraints do not have to be satisfied for a feasible solution; however they should be obeyed if possible to make a more qualitative schedule. The soft constraints usually represent special policies of the institutions and thus have to be set individually in each special case, as well as require a certain degree of experience of people who use the schedule. (Lewis, 2007) The examples of soft constraints include teacher's preference for morning classes, lack of large time windows between classes or scheduling only one exam per day.

Additionally to hard and soft constraints McCollum et al. (2007) introduce a third highest level of constraints as a so-called required constraints. In their formulation, the required constraints should be absolutely non-violated, whereas violation of hard constraints is possible but leads to a "non-zero distance to feasibility" (p.6). Further, McCollum et al. (2007) argue that the quality of any timetable depends on the number of hard constraint violations and amount of implemented soft constraints. Thus the lower the distance to feasibility is, the better the schedule is. At the same time, the large the number of respected soft constraints, the more feasible and practically usable the schedule becomes.

Another generalisation that can be outdrawn about scheduling problems is that in almost all cases they are not solved polynomially (except for simple school timetabling problems, see Chapter 2.1). Therefore, most of the scheduling problems are NP-complete. Due to the complexity of the problems, a search for an optimal solution is not always a necessary task. Instead for practical rationale the approximation algorithms that provide at least a feasible solution are mostly being used. (Lewis, 2007)

A wide range of solving approaches is applied to solve NP-complete scheduling problems; some of the main techniques are discussed in detail further in this paper. Nonetheless, before the discussion takes place, it is possible to divide the

approximation algorithms into three main subcategories, which can be applied for the most of the scheduling problems (Lewis, 2007, p. 7):

1) One – Stage Optimisation Algorithms, which aim at simultaneous solving of hard and soft constraints (cf. Abramson et al., 1996; Schaerf, 1996; Carrasco & Pato, 2001; Di Gaspero & Schaerf, 2002);

2) Two – Stage (or Multi-stage) Optimisation Algorithms, which satisfy first hard constraints, while soft constraints are considered only in case of existing feasible timetable from the first stage (cf. Pillay & Banzhaf, 2009; Malik et al., 2010; Qaurooni et al., 2011; Gogos et al., 2012);

3) Algorithms that allow Relaxations, which prohibit the violations of the hard constraints first by trying to lose other elements of the problem and secondly by removing soft constraints relaxations where possible (cf. Bateburg & Palenstijn, 2003; Daskalaki & Birbas, 2005; Pongcharoen et al., 2008; Burke et al., 2009)

Finally, in order to organize the great amount of different timetabling and scheduling problems it is common to systematize them into subgroups by their main goal or objective function. Therefore, a large family of problems can be classified into three main classes (Schaerf, 1999):

1) School timetabling problems aim to assign classes to time periods, where the goal is to prohibit involvement of a teacher into two simultaneous classes at a time.

2) (University) course timetabling considers a weekly scheduling of lectures from a set of courses with the aim of minimizing the number of common students in the various lectures at the same time.

3) (University) examination timetabling schedules the exams to the set of courses taking into account that a student should not be assigned to a several exams at a time.

Further, this chapter describes each of the above mentioned timetabling problems more detailed, including short discussion of the solution methods proposed in the literature.

2.1 SCHOOL TIMETABLING

School timetabling problems are also known as class/teacher models. Those problems are the most uncomplicated to solve among all of the above mentioned models from timetabling family. Although most timetabling problems are NP-hard, the basic or simplified school timetabling problems can be still polynomially solvable (Eikelder & Willemen, 2001). Yet, simplified models are seldom applicable to the reality. Indeed, depending on the requirements of the institutions high school problems can represent quite complicated problems as, for instance, the school schedule has to be continuous in comparison to the university schedules. Another reason for complexity is the problem's size, which grows proportionally to the number of students. Thus the larger the problem, the more difficult is the satisfaction of the conflicting constraints. (Zhang et al., 2003) Consequently, creating an indisputable schedule is not an easy task.

The basic polynomial school timetabling problems have been proposed by de Werra (1985) where the aim is to assign lectures to periods under the assumption that no teacher or class is scheduled to more than one lecture at a time. This problem is formulated as a search problem that should find a feasible timetable that satisfies all constraints. The constraints in this simplified polynomial problem ensure that each teacher gives the pre-defined amount of lectures to each and only one class at a time.

Solution techniques

According to Even et al. (1976), if the number of lectures is set to a definite value, the problem runs in the polynomial time. To find a solution to this problem, a bipartite multigraph matching should be solved, whereas a class and a teacher are viewed as vertices, and the links between them as edges.

Another solution to a problem would be an edge colouring method on graphs (s. Figure 2-1). This method considers each event as a vertex and then finds a link between them and marks it as an edge. By associating each period (or a timeslot) with a colour, the method finds an assignment of a colour to each edge in a way that none of the neighbouring edges is coloured the same. (de Werra, 1985)

The graph colouring method can also be used for solving NP-hard problems. In this case additional requirement should be observed. Namely, a solution should not

use more colours than there are available timeslots or the number of colours used should be kept minimal. (Lewis, 2007)

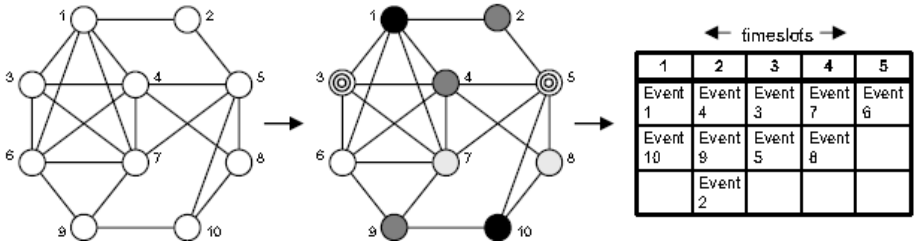


Figure 2-1: Graph colouring solution to a simple timetabling problem.

Source: Lewis (2007), p. 5

The Figure 2-1 provides an example of a simple timetabling problem, where 10 lectures (i.e. 10 colour vertices) have to be assigned. The solution is then found by assigning the optimal number of colours (i.e. 5 colours for 10 vertices). As it is seen from the graph, none of the connected vertices are coloured the same. Finally, the graph colouring problem is converted back to a timetabling problem, where each colour is associated with a timeslot. For example, black vertices represent a first timeslot. This method proves that none of the conflicting events (i.e. adjacent vertices) are scheduled to the same timeslot. (Lewis, 2007)

Although, the graph colouring solution method is an easy and fast technique, it is usually applicable only when simple constraints have to be followed. In the most cases, the timetabling problems have a large number of more complex constraints, e.g. simultaneous lectures for classes (Yoshikawa et al., 1996), no time windows between the classes (Schaerf, 1996), special rooms such as gym, science lab etc. (Schaerf, 1999)

Moreover in the real application a school timetabling problem is usually formulated as an optimization problem, with the goal to minimize the number of non - desirable timeslots for a teacher (Junginger, 1986) or on the contrary to maximize the number of assigned desirable lectures for a particular teacher (de Werra, 1985).

The solution methods for NP-hard school timetabling problems include:

- Direct heuristics (cf. Junginger, 1986; Schmidt & Strohlein, 1979)
- Genetic algorithms (cf. Ross et al., 2003; Burke et al., 2009; Raghavjee & Pillay, 2012)

- Tabu-search (cf. Hertz, 1992; Tassapoulos & Beligiannis, 2012)
- Neighbourhood search (Zhang, Liu et al., 2010; Brito et al., 2012; Post et al., 2012)

2.2 COURSE TIMETABLING

The university course timetabling problem schedules a set of lectures for each course into a weekly timetable under restrictions concerning given number of rooms and time periods. The main difference of course and school timetabling problems is that in school problems the room availability is often neglected, as school classes mostly have one constant room for all courses. In addition, school teachers participate in many different lectures taught more than just to one class, whereas university professors are usually responsible for one course only. (Schaerf, 1999)

Feasible solution is reached when all lectures were set in accordance to a period and rooms, as well as the hard constraints were satisfied. (Di Gaspero et al., 2007) The most important hard constraints are following (Lu & Hao, 2010):

- Two different courses cannot have common students at the same time period
- A room cannot be shared by different courses at the same time period
- A teacher should be scheduled to only one lecture at a time period
- A student (group of students) cannot have overlapping lectures at the time period.

Therefore, a feasible solution considers different room requirements, whereas in practice under the definition of rooms not only actual classrooms are being classified, but also labs, gyms, offices, planes or any other possible studying facility. Thus, university course timetabling problems can be applied to a much wider scope of problems beyond university scheduling.

The course timetabling problems can be distinguished into two categories: the master timetable (or curriculum-based) and demand driven systems (or post-enrolment). The master timetable first creates a timetable without consideration of students' preferences. The students choose available lectures from already existing timetables, thus schedule their personal timetables themselves and ex-post. (Alvarez-Valdes et al., 2002) The Vienna University timetable system is an example of the master timetable. The demand driven system allows the students to choose first from

the number of available courses, which are then compounded into a feasible timetable with time periods, thus the individual preferences of the students are important. (Carter & Laporte, 1997) This paper represents an example of demand driven systems, where the timetable is created after all students have claimed their available time periods in the upcoming month.

In order to make timetables more usable and better in practice, a number of soft constraints might be considered. As soft constraints are significantly more practical than hard constraints, their exact number and formulation depends on the university requirements. The most used soft constraints, as well as preferences that are important in this paper include:

- Timetables should be as compact as possible, without lengthy breaks (i.e. idle times) between several lectures in a day (Alvarez-Valdes et al., 2002)

- Timetables should respect preferences of teachers regarding their teaching days, times or amount of consecutive periods. In addition, the teaching load shall correspond to their set amount of working hours per week and should be balanced with other teaching stuff. (Zhang et al., 2010)

- The timetable should use any extra rooms, equipment, teaching stuff or similar only if all existing common resources are already assigned

- Timetable should respect the sequence of courses if it exists, and thus not assign advanced lectures to beginner classes (students), thus the lectures should be taken in a consecutive order (Lu & Hao, 2010). Moreover, each course should consists of exact amount of lectures needed to pass it (Schaerf, 1999)

- Well-designed timetable should adjust the capacity of the learning facilities to the desired number of students, thus rooms shall neither be overcrowded or incomplete (Alvarez-Valdes et al., 2002)

- Room stability constraint ensures that all lectures of a particular course take place in the same room (Lach & Lübbecke, 2012)

Similarly to individual choice of soft constraints, the selection of the objective function varies according to the model structure and/or university preferences. Some researchers (Lewis, 2007; McCollum et al., 2007; Zhang et al., 2010; Lu & Hao, 2010) propose to set penalty weights to the constraints, which influence the overall quality of the timetable. Afterwards the total amount of penalties should be minimized. Moreover, the violations of hard constraints set the quality of the timetable to very low and lead to "zero distance to feasibility" (McCollum et al., 2007, p.6), whereas violations of soft constraints are possible. The individual quality of the timetable depends on the amount of allowed soft constraints' relaxations.

Less popular composition of the objective function for the university course timetabling problem is to maximize the resemblance of preferences (e.g. teacher or student) for course or period combinations with the actual allocation of the feasible timetable (Schaerf, 1999; Lach & Lübbecke, 2012). This objective function is especially useful in a demand driven system, as the pre-given preferences regarding lectures or individual available timeslots of the students are subject to construct and maximize in a latter creation of a valid timetable. Therefore, further in this paper this type of objective function is going to modelled, modified and applied to the real-life problem (s. Chapter 4.2).

Solution techniques

As mentioned above, most of the timetabling problems are NP-hard. Due to the fact that timetabling problems are well-researched, a various number of solution technique exists. One of the first proposed solution methods is a reduction to graph colouring proposed by de Werra (1985) already 30 years ago. Graph colouring technique is explained in details on an example of school timetabling problem earlier in this paper (see Chapter 2.1).

Another quite common way of dealing with the timetabling problems is an integer linear and mixed programming technique. The problems can be either formulated as assignment problems or as a 0-1 optimization problem. Other tools include Lagrangian Relaxation to formulate the problem as a transportation model (Miyaji et al., 1987) or to assign classes to rooms (Laporte & Desroches, 1986; Carter M., 1989; Tripathy, 1992; Daskalaki et al. 2004).

Further on, the network flow model can be also used for solving course timetabling problems. For example, Dinkel et al. (1989) suggest to create a network

model with three levels, where the first level is responsible for a vertex of departments, the second level holds a vertex for teaching stuff and course matching, and the third level is a vertex holding an information about the room and time period combination. The problem is then solved by modelling a penalty function, while approaching to optimize the network. Authors state that very large problems can be solved easily by a network algorithm. However, conflicts concerning the simultaneous assignment of a teacher to different courses may appear.

Although the above mentioned methods provide a feasible solution, more advanced techniques are usually applied in practice. Foremost, local search heuristics have gain large popularity in the last decades. In general, the local search approach aims at finding first an initial solution and then continuing of choosing appropriate neighbourhood moves for a better solution. Nonetheless, the process of comparing and optimizing all of the moves might be quite lengthy. (Jaziri, 2008)

One of the most popular local search algorithms is a Tabu Search method. Tabu Search moves from a current solution to a neighbouring solution keeping the finite list of visited neighbours. On an example of course timetabling problems, the moves would consider the re-ordering of time periods of lectures. Therefore, the explored moves are not repeated, improving this way the performance of the algorithm. However, in order to find a feasible some the hard constraints should be relaxed. Nevertheless, to keep hard constraints, Schaerf (1996), for instance, proposes to insert two neighbourhood operators, whereas one is commonly looking for neighbourhood moves, but the second operator moves with the aim to eliminate any appearing conflicts of the first move.

Another one stage optimization algorithm is a simulating annealing. In comparison to Tabu Search, simulating annealing techniques do not memorize the previous moves, yet decide through a system of adoptive cooling if a next move should be made. One drawback of this method is that it may happen that a move will be made, even if it is not better than a current solution. To avoid this Elmohamed et al. (1998), for example, propose to model a weighted-sum scoring function, whereas hard constraints penalties are notably higher than those for the soft constraints. By using a function and through simultaneous setting of an adaptive cooling, good starting solution, as well as smart methods for choosing next moves good results can be achieved.

The third well used Local Search technique is genetic algorithms. In comparison to two of the above mentioned algorithms, genetic algorithms are able to hold many individual solutions at a time in form of a population. The algorithm reflects a process of the transfer of genetic information from parents to a child. Parents form a child, which has different attributes of both parents, as well as his own personal characteristics. Thus, through a mutation, the child becomes new genes. The better the genes of a child, the more fit is it to survive in a population. (Abramson & Abela, 1992) Just like in the nature, the strongest survives, producing thus the best optimal decision. In a timetabling problem, the child with the strongest genes is a timetable, which does not violate any of the hard constraints, although his parents may have violated one. Genetic algorithms are applied more often to university examination problems than to course timetabling.

Another group of techniques can be classified as logic programming. One of the first logic programming methods was proposed by Fahrion & Dollansky (1992). In order to solve timetabling problems they suggest using PROLOG language, which aims to model the constraints declaratively based on a priority of lectures to be assigned into a schedule. Constraint logic programming techniques set the basic components of the problem as constraints. The logic matching of the constraints by a specific rule and through a particular algebraic semantics is then a goal of the whole problem. (J. & Lassez, 1987) The logic scheme can be written in any available programming language (e.g. PROLOG, LOGIN, Mosel, CHIP, C++, Java).

Finally, the more modern technique – constraint programming (CP) allows more flexibility for formulating the timetabling problem by allowing relaxations by over-constrained problems (Cambazard et al., 2004). An initial formulation of a CP is a set of variables with a domain and a set of constraints for each of the variables. A feasible solution is a turn of each variable into a value, whereas all constraints are satisfied. (Hahn-Goldberg, 2007) As relaxations are possible, the quality of the optimal solution depends on a number of actually satisfied constraints. A search for the best solution is done in form of a tree approach, whereas different heuristics exist to build a feasible tree.

While local search techniques are notably more used than CP models, some authors believe that CP can provide good solutions, especially in a combination with other methods (Cambazard et al., 2012). Moreover, most of the solution methods for the timetabling problems in the last years suggest applying a number of diverse

techniques simultaneously (cf. a review by (MirHassani & Habibi, 2013). Such approaches usually solve the problem in multiple stages, whereas first the large problems are decomposed into smaller ones, secondly the hybrid heuristics search methods are applied, and finally the feasible solution for the initial large problem is found. The sophisticated techniques include among others the use of meta-heuristics (cf. a review by Lewis (2007)), the hyper-heuristics (De Causmaecker et al., 2009) or ant-colony algorithms (Socha et al., 2003).

As mentioned above, the application of IP, MIP or CP techniques to course timetabling problems was not much favoured in the past due to their slowness and inability to solve NP-hard problems in real-time. However, according to MirHassani & Habibi (2013), in the recent years IP and MIP formulations have again gained popularity in an academic research and practical applicability due to the development of the informational systems and advanced software tools, which are able to solve timetabling problems with thousands of integer variables and millions of binary variables in a short time.

This paper also focuses on modelling the course timetabling problem in form of MIP formulation and researchers whether medium problems can be solved easily with the help of CP techniques. It is then programmed in Mosel language and solved with the help of the Xpress software (see Chapter 4&5 for detailed results).

2.3 EXAMINATION TIMETABLING

The university examination timetabling problems have the task to assign a set of exams into timeslots and rooms subject to a number of constraints. An exam should be thus scheduled for each existing course. Course and examination timetabling problems are quite related, yet some differences exist. Foremost, the conflict between exams is significantly stricter than between lectures, as student may skip an overlapping lecture but not an exam. Furthermore, the exam periods are not fixed and may vary. Finally, constraints in an examination timetabling usually alter from course timetabling. (Schaerf, 1999)

A large variety of hard and especially soft constraints exists in the literature. As some constraints are similar to course timetabling problems, yet others are unlike, the primary hard constraints (Table 2-1) and soft constraints (Table 2-2) are provided in the tables below. Constraints illustrated in the tables below are mainly based on the survey papers of McCollum et al., (2007), Qu et al., (2009) and MirHassani & Habibi

(2012), who have contributed the most in the recent years in summarizing, grouping and comparing a remarkable amount of academic papers done in the field of timetabling problems.

Table 2-1: Primary hard constraints in examination timetabling problems

Primary hard constraints
1. Students are not assigned to two exams simultaneously (Qu et al., 2009)
2. Facility resources should match student demand (i.e. enough rooms for all; number of seats not less than number of students) (Qu et al., 2009)
3. Every exam is allocated to a particular room (or particular rooms if number of students exceeds capacity of one room) (McCollum et al., 2007)

Table 2-2: Frequently used soft constraints in examination timetabling problems

Frequently used soft constraints
1. A room cannot hold more than one exam simultaneously (Causmaecker et al., 2009)
2. Conflicting exams should be spread as evenly as possible. (Qu et al., 2009)
3. Schedule multiple exams of the same duration (or of the different duration) simultaneously into the same room (McCollum et al., 2007)
4. Exams should be consecutive or more advanced exam should follow the beginner level. (Qu et al., 2009)
5. Assign large (complicated) exams as early as possible into a timetable (Qu et al., 2009) or as late as possible (McCollum et al., 2007).
6. Aim to assign all of the exams of a particular subject to one room. (Qu et al., 2009)
7. Regard teachers' preferences towards room facilities, equipment or timeslot. (MirHassani & Habibi, 2013)
8. Avoid assigning student to two or more exams per day. (McCollum et al., 2007)

In addition to hard and soft constraints, researchers introduce diverse opportunities for the design of the objective function. Whereas Schaerf (1999) proposes to find a feasible solution that does not aim to minimize or maximize any given values of the problem but rather has a goal to provide any working timetable that satisfies hard constraints, other researchers suggest setting penalties for the violations of the hard or soft constraints. Table 2-3 provides a short overview of the possible objective functions.

Table 2-3: Primary objective functions in the examination timetabling problems

Frequently used objective functions
1. Find a feasible timetable that satisfies all hard constraints. (Schaerf, 1999) 2. Minimize penalty weights for the room, period or student/teacher mismatching. (McCollum et al., 2007) 3. Maximize interests or preferences of the students/teachers. (Dimopoulou & Miliotis, 2001)

More information about the design of the examination, as well as course and high-school timetabling models can be found on the web pages of First[1] (2002), Second[2] (2007) or Third[3] (2011) Timetabling Competitions. The aim of the competitions was to stimulate timetabling research and attract researchers to develop theoretical models, which are closely interrelated with practice. To achieve that real-world practice problems were supposed to be solved either as quickly as possible or with the best practical (feasible) results. More specifically, the First Competition that focused largely on course timetabling aimed to create a common ground for comparison and grouping of diverse amount of existing literature contributions in timetabling field. The Second Competition had a goal to subdivide timetabling into three tracks (both types of course timetabling and examination problems) and stimulated more practically oriented research. Finally, the Third Competition focused on high school timetabling, whereas proposed real-world problems were of NP hard type in order to encourage complex research findings.

Solution methods

The solution methods for examination timetabling problems are in general the same as for school and course timetabling ones (see Chapters 2.1; 2.2). For a complete review of solution methods it is useful to become familiar with the review papers of Carter & Laporte, (1997), Carter, (2001) and more recent surveys by McCollum et al., (2007), Qu et al., (2009) and MirHassani & Habibi (2013). Moreover, for a more detailed research in the field of the timetabling problems author of this paper suggests a study of the numerous works of E.K. Burke and his co-researchers (cf. 1997, 2004, 2007, 2009).

[1] http://www.idsia.ch/Files/ttcomp2002/
[2] http://www.cs.qub.ac.uk/itc2007/
[3] http://www.utwente.nl/ctit/hstt/itc2011/welcome/

3. PROBLEM BACKGROUND

The aircraft daily scheduling problem developed in this thesis is based on the demand driven (or post-enrolment) course timetabling model, as it was already mentioned in the previous chapter. This thesis aims to convert theoretical concepts to a real – life situation by modelling a timetable for a flight school. The idea to create a timetable for a flight school was motivated by and grounded on the real scheduling problems of the JA Flight Training GmbH.

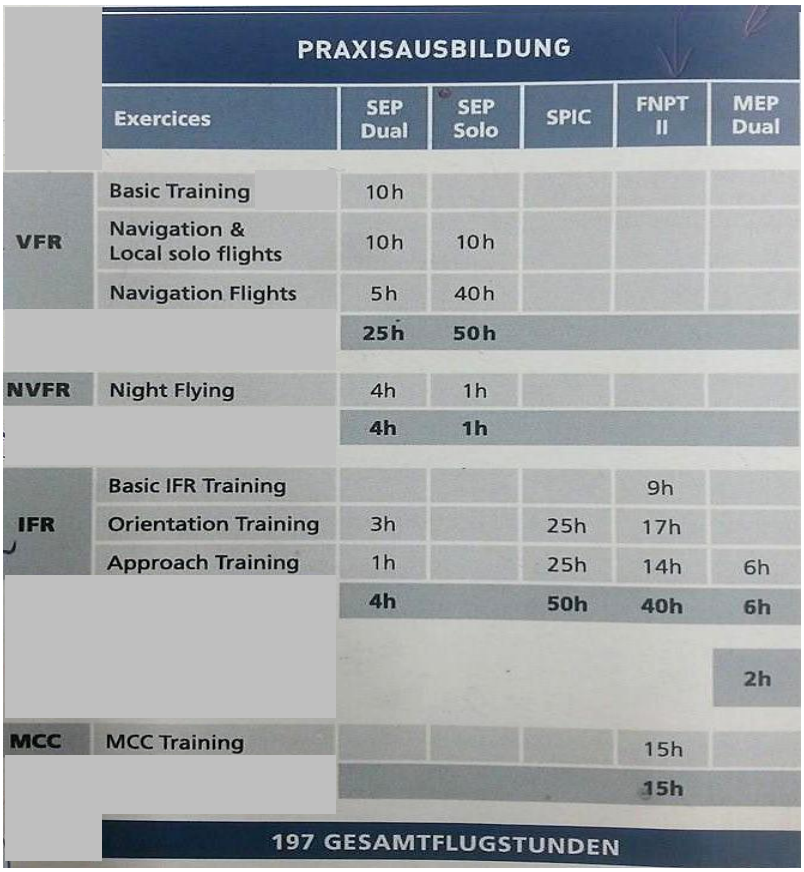
JA Flight Training GmbH was a branch enterprise of Jet Alliance Group and was located 30km away from Vienna, in Kottingbrunn, Austria. During the time span of an elaboration of this thesis, the company was forced to a bankruptcy because of the financial difficulties of Jet Alliance Group in the beginning of December 2013 (Metzenbauer, 2013). Nonetheless, the simulator centre and infrastructure of the JA Flight Training was recovered by Aviation Academy Austria, who took over the company on the 3rd of March, 2014 (Metzenbauer, 2014). Therefore, further in this paper, the details about the company are stated in the past tense. Nonetheless, the development of the model remains interesting for a research despite the fact that the real company does not exist up-to-date. The reasons for problems' actuality are following: (1) the company requirements are common for any other real flight training company, thus (2) the model developed in this thesis can be applied to any existing flying school or (3) any other planning problem of an educational small-to-medium sized business.

3.1 BASIC SETUP

The company had one of the most modern and high-qualitative aviation centres in Austria. JA Flight Training offered a wide-spectrum of educational courses and trainings for pilots and crew. The courses included theory lectures, practical flying trainings on planes, as well as flying hours on a modern plane simulator. The complete training from zero to ATPL frozen[4] (Air Transport Pilot License) prepared absolute beginners to become pilots on a commercial airline up to 9 seats. Duration of the ATPL course ranged between 12 and 36 months, whereas in practice full-time studies were finished in approximately 13-18 months and part-time in 18-30 months. (JA Flight Training, 2013)

[4] ATPL frozen license becomes "unfrozen" after flying 1500 hours as pilot in command after completing the full ATPL course

The common ATPL course starts with theoretical lectures, which are obligatory for all participants. Theoretical courses are planned in advanced without consideration of students' personal preferences. Thus, students are personally responsible for managing their personal schedules according to pre-given lecture times. The lecture times are usually similar each year and do not need any special planning as historical demand does not tend to overweight supply of learning facilities (e.g. lecture rooms) and the availability of teachers.



PRAXISAUSBILDUNG						
	Exercices	SEP Dual	SEP Solo	SPIC	FNPT II	MEP Dual
VFR	Basic Training	10h				
	Navigation & Local solo flights	10h	10h			
	Navigation Flights	5h	40h			
		25h	**50h**			
NVFR	Night Flying	4h	1h			
		4h	**1h**			
IFR	Basic IFR Training				9h	
	Orientation Training	3h		25h	17h	
	Approach Training	1h		25h	14h	6h
		4h		**50h**	**40h**	**6h**
						2h
MCC	MCC Training				15h	
					15h	
197 GESAMTFLUGSTUNDEN						

Figure 3-1: Structure of ATPL practical hours.

Source: JA Flight Training (2013): Ausbildungsprogramm (p. 12)

After first theoretical lectures were completed, a student may start with the practical lessons. Practical flying hours are divided into two parts: actual airplane flying (SEP, SPIC, MEP) and synthetic flying on the full-flight simulator (FNPT) (see Figure 3-1). Synthetic flight training consists of 55 obligatory hours per student, airplane flying hours equal to 140 hours per student. Students are allowed to choose individually how many flying hours each month they are willing to complete. Moreover, as most of the students are enrolled for the ATPL course part-time only, the flying school gives them an opportunity to select dates for their individual flying trainings.

Therefore, by the end of each month students are impelled to send via e-mail their preferred dates and amount of flying hours for the coming month to the school. After the preferences were received, schools' management starts to plan a monthly timetable. Besides students' preferences, a number of other requirements have to be obeyed as well. For instance, a good timetable should also respect teachers' choices that may monthly vary (e.g. morning/evening hours, maximum amount of working hours per week, different airplane licenses), airplane characteristics (e.g. matching with student's and teacher's skills), weather conditions and further factors.

The biggest complexity in a timetable planning is a fact that up to this time point a flying schedule was created manually. Manually generated timetable often leads to an unsatisfying "is" situation. Figure 3-2 compares the "is" situation with the "should" planning condition. To change the "is" situation to the "should" state, this paper proposes a model for an automated timetabling system, which will create auto-generated timetables with less effort and in significantly shorter time.

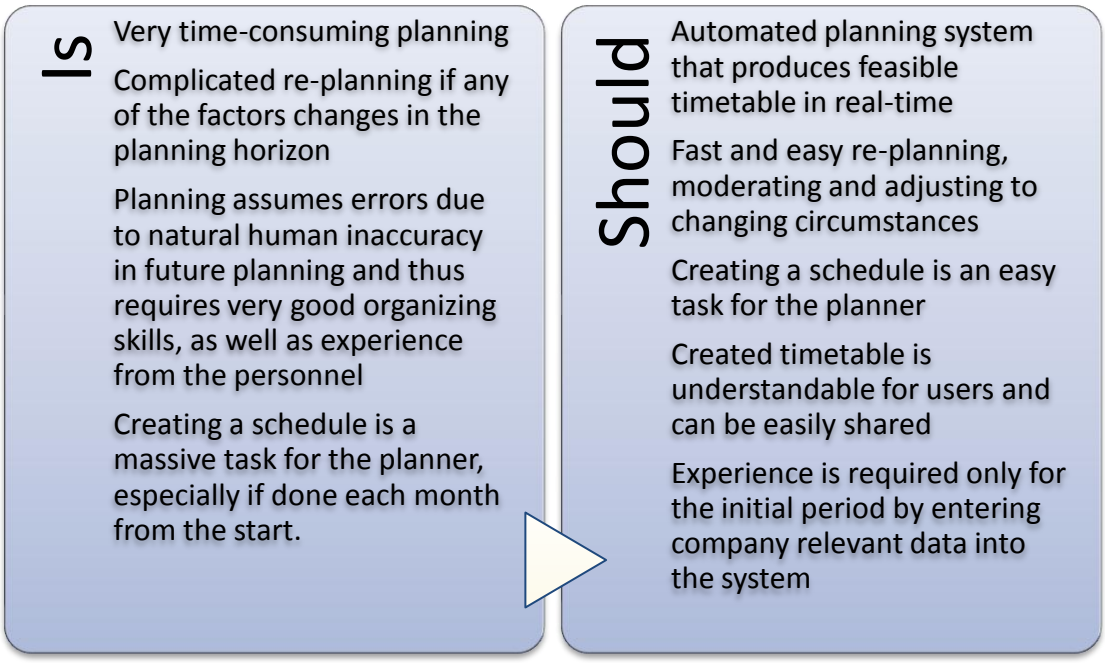


Figure 3-2: Comparison of "is" and "should" planning situation

A new computer-aided system will generate monthly timetables automatically with the minimal manual input. Namely, a planner will only have to insert available times of the students into the program. The basic details about teachers' preferences and skills, as well as airplane characteristics and course descriptions will be set in advance. The goal of the system is then to match student's time preferences with the corresponding course, its teacher and airplane and to assign him/her to a timeslot.

Moreover, no other student can be assigned to the same timeslot with the same teacher and/or plane. In this way, the model avoids inconsistencies, which are quite possible by manual planning. Therefore, the system will create a timetable with a consideration of all requirements.

Furthermore, the model developed in this thesis helps the timetable planner to answer following questions:

- Which and how many timeslots are scheduled per month? What is an optimal amount of scheduled timeslots per month?
- Which teacher should be assigned to a particular student/timeslot/plane?
- Is it profitable to employ external flying instructors if students' demand or preferences exceed monthly capacities?
- Which plane should be chosen for the optimal monthly profitability?
- Is it profitable to rent external facilities (i.e. planes) if monthly demand excels supply?

It is clear that the above-stated questions are not new and have been well elaborated in the timetabling literature. However, the model proposed in this thesis does not aim for the novelty but rather for the practical implementation of real-world problem. In addition, the model aims to connect timetabling problems with the aircraft scheduling with a consideration of specific requirements in the aero field, whereas most usually all practical realizations of timetable problems were proposed in the classical school/university scope.

As mentioned before, the problem will be mathematically modelled as a MIP formulation and then programmed in Mosel language (see Chapter 4&5). Nonetheless, the mathematical model can be also programmed in any other programming language, which is applicable for mathematical modelling, and then be solved by a number of available commercial solvers.

3.2 DESCRIPTION OF THE REQUIREMENTS

This chapter describes the requirements for the model in a more specific detail. The purpose is to introduce the reader to particular needs of a timetable in a flying school, while considering common elements of general timetable problems as well. To make the problem as realistic as possible, the model requirements illustrated

below are based on the real issues faced by JA Flight Training GmbH. Namely, following requests have to be followed in the model:

1. A timeslot equals to one astronomical hour (60 minutes).

2. An input of the model is all possible timeslots of the month, which are reported monthly by e-mail by each student.

3. The output of the model is the allocation of timeslots to students, whereas each student should be connected to one teacher and one plane. In an output view, the program should also show in which studying hour the student currently is and which course corresponds to this hour.

   Table 3-1 shows an example for one student for the first two timeslots of the month, including data needed for planning (input), as well as produced by the model (output).

Table 3-1: Input and output data of the timetable model

	Student "Example"			
	Input	**Output**		
	Availa ble:	Assigned to:	Has completed (hours):	Is in course (#):
Timeslot 1 (Mo, 23.3, 9-10)	Yes	Mr. "Rall" Plane "DA42"	10	1
Timeslot 2 (Mo, 23.3, 10-11)	Not	0	11	2

Basic constraints

4. Each student in each timeslot can be assigned only to one teacher and one plane.

5. Each plane in each timeslot is assigned only to one student and one teacher.

6. Each teacher in each timeslot should be assigned to only one student, if and only if he is flying with him. Otherwise, if teacher is observing the students from the ground, who are practicing flying alone, he can be assigned to maximum 3 students at once.

Case-based constraints

7. A school is opened from Monday till Friday from 9 am to 9 pm, whereas sunlight (day time) hours last from 9 to 18 and after twilight hours (night time) last from 18 to 21.

8. A student is not allowed to fly more than 6 hours in a day. After the 6 hour block time, student has to take a rest period for at least 8 hours. In addition, it should be considered that 3 last hours of a working day and the first 3 hours of the following day are not considered as 6 consecutive hours of block time (Table 3-2).

Table 3-2: Example of a valid and invalid timetable according to block and rest time criteria

Students [1..i]	1	2	…	9	10	11	12	13	14	15	A rest time of minimum 8h is ensured between timeslot 12 and 13 – a valid timetable
	9-10	10-11	…	17-18	18-19	19-20	20-21	9-10	10-11	11-12	
1	1	1	…	1	1	1	1	1	1	1	
i	0	0	…	0	0	1	1	1	1	0	
		Timeslots [1…15]									
		Hours									

Students [1..i]	1	2	3	4	5	6	7	…	14	15	A block time of more than 6h in one day without rest time – an invalid timetable
	9-10	10-11	11-12	12-13	13-14	14-15	15-16	…	10-11	11-12	
1	1	1	1	1	1	1	1	…	0	0	
i	0	0	0		0	1	1	…	1	0	
		Timeslots [1…15]									
		Hours									

9. A teacher is not allowed to work or fly more than 12 hours in a day.

10. A full-time teacher works 40 hours in a week.

11. Private instructors work independently and are paid extra for each service. Therefore, their services are more expensive and should be used only if no full-time instructor can hold a lesson for a student.

12. Each instructor may only teach the lectures for which he is certified. The problem assumes that one teacher is capable of teaching beginner classes only (i.e. the first 6 courses), whereas other two full-time teachers can hold lectures in all courses.

13. The model will only concentrate on actual flying hours, as requested by the company. Creating the timetable for the airplane flight training is the most critical and time consuming task in practice.

14. The complete ATPL actual flying lectures (see Figure 3-1 for initial structure) can be divided into 11 courses with different contents. Each course has different duration, as well as plane, student and teacher skills requirements. The courses have to be completed in an increasing consecutive order, or in other words, one after another. Table 3-3 shows a detailed description of the courses and their characteristics.

Table 3-3: Characteristics of the aeroplane flight training courses in scope of ATPL program

Course name (q_course)	Duration of the course (Q)	Hours completed (β)	b_n^i– parameter for instructor location (1=air; 1/3= ground)	Plane type
K1 = Basic	10	[0; 10]	1	all
K2 = Navigation	10	[11; 20]	1	all
K3 = Local solo	10	[21;30]	1/3	all
K4 = Dual cross-country	5	[31;35]	1	all
K5 = Solo cross	40	[36; 75]	1/3	all
K6 = Night Dual	5	[76; 80]	1	all
K7 = Orientation Dual	3	[81;83]	1	all
K8 = Orientation SPIC	25	[84; 108]	1	all
K9 = Approach dual	1	[109]	1	all
K10 = Approach SPIC	25	[110; 134]	1	all
K11 = Multi engine	6	[135;141]	1	DA42

As seen from the Table 3-3, planes from type C172S and C172R can be used only in the first 10 courses, whereas C172S should be forced to be assigned before C172R due to its lower operation costs and higher suitability for beginner students. Plane type DA42 is the most expensive plane that has a multi-engine piston and is the only type of aircraft that can be used in all courses including course 11.

Further, Table 3-3 also illustrates that course K6 is a night flying training and thus should be only assigned to an evening timeslots, which start at 18:00 and finish at 21:00.

15. If a student has completed 141 flying hours ($\beta = 141$), he has completed the studies and no timeslots should be assigned to him anymore.

Further modifications and possible improvements:

16. As an improvement to the model, which is however not obligatory, the change of twilight hours according to the month/season can be considered.

17. In the ideal situation, an instructor should consider 30 minutes break between the releases while observing students from the ground. In other words, a second student can only depart in 30 minutes after the take-off of the previous student.

18. The flying is only possible in the good weather conditions. The model could also consider different weather forecasts that could be implemented in the model as a probability function. Basing on the degree of the probability (e.g. if the probability of the bad weather is high), the program should re-calculate and provide further feasible assignments of students to timeslots.

4. MATHEMATICAL MODEL

This chapter introduces a mathematical model, which was developed in scope of this thesis. First of all, a notation that is used in the model is described. Further, an elaboration of the mathematical formulation follows, which focuses on the (flight) course timetabling modelling.

4.1 NOTATION

First of all the following sets have to be defined:

Let N be the set of timeslots, whereas each timeslot n = 1…N represents one astronomical hour or 60 minutes. Set of flight instructors (teachers) is defined by J (j = 1…J). Set I (i = 1…I) holds information about currently participating students. Fourth main set Z (z = 1…Z) consists of the planes and their characteristics. The set Q (q = 1…Q) defines groups of courses and their characteristics. Table 4-1 summarizes the data sets used in the model.

Table 4-1: Notation for the sets used in the model

Abbreviation	Description
N	Set of timeslots
J	Set of teachers
I	Set of students
Z	Set of planes
Q	Set of courses

The model further includes parameter (input) values. Table 4-2 shows the notation used for the parameters in a summarized view. First of all, the availability matrix of timeslots should be added. The availability matrix shows timeslots (or a number of successive timeslots), which the student is able to attend in the upcoming month. This parameter is inserted manually by a planner into the program and is denoted as $\mathrm{AVAL}^{i}_{n} \in [0,1]$, whereas $\mathrm{AVAL}^{i}_{n} = 1$ if student i is available at timeslot n, and $\mathrm{AVAL}^{i}_{n} = 0$ otherwise.

The next two parameters are denoted as C_PLANE_Z and $C\_INSTRUCTOR_J$ and represent different costs for planes and instructors, whereas the costs for internal teachers are lower than for external instructors, and simpler (or older) planes are cheaper than more advance ones. Parameter P stands for the profit of each assigned

timeslot and is calculated as a difference between revenues (costs for education paid by students) and costs (i.e. operation costs for plane, salary for teachers).

Parameter PH^i illustrates the amount of previously completed hours by each student from the preceding month. Parameter S sets the maximum amount of consecutive flying hours, which are allowed per day. MAX_N stands for the maximum amount of timeslots per month, which depends on the working hours of a school. The last parameter used in the model indicated as FULL_P sets the total amount of hours, which are needed for each student to complete the full studying program.

Table 4-2: Notation used for the parameters in the model

Abbreviation	**Description**
$\mathrm{AVAL}^{\mathrm{i}}_{\mathrm{n}} \in [0,1]$	Student i is available at timeslot n
$\mathrm{C\_PLANE}_{\mathrm{z}}$	Operating costs for airplanes
$\mathrm{C\_INSTRUCTOR}_{\mathrm{j}}$	Costs for teachers
P	Profit
PH^i	Previously completed hours by student i
S	Maximum amount of consecutive flying hours
MAX_N	Maximum amount of timeslots per month
FULL_P	Full amount of studying program hours

Table 4-3 shows the decision variables used in the model. All of the decision variables are important for the output of the model.

Table 4-3: Notation used for the decision variables in the model

Abbreviation	**Description**
$x^i_{njz} \in [0,1]$	Assignment of student i to a timeslot n, teacher j and plane z
β^i_n	Completed hours up to timeslot n of student i
$q\_course^i_n$	Course number of student i in timeslot n
$b^i_n \in \left[1; \frac{1}{3}\right]$	Location of a teacher [air; ground]
y^i_{njz}	Product of $b^i_n \times x^i_{njz}$
$d^j_n \in [1,0]$	Assignment of teacher j to a timeslot n

The decision variable $x^i_{njz} \in [0,1]$ denotes an assignment of student i to a timeslot n, teacher j and plane z. If the student was assigned to timeslot n, the

variable takes value $x_{njz}^{i} = 1$, else $x_{njz}^{i} = 0$. β_n^i calculates current studying hour up to a timeslot n of student i. Decision variable $q\_course_n^i$ shows a number of the course, in which student i is enrolled at a timeslot n. Note that this decision variable serves for illustration purposes of the timetable only and is not directly participating in the process of searching for the optimal solution.

Further decision variable b_n^i denotes the location of a teacher during the lecture, depending on the course requirements or also on the value of β_n^i . If the teacher is flying with the students, when $b_n^i = 1$ as a lecture can be given to only one student i at a timeslot n. On the other hand, if the student i is having a solo flight at a timeslot n and is being observed by a teacher j from the ground, when $b_n^i = 1/3$ as a teacher j can observe up to three students in their solo flight courses.

Decision variable y_{njz}^{i} is used for calculation purposes only due to the need to transform a non-linear problem with a multiplication of two decision variables into a linear formulation. Therefore, y_{njz}^{i} denotes a product of b_n^i and x_{njz}^{i}. A more detailed explanation follows in the next chapter (Chapter 4.2). The last decision variable d_n^j denotes an assignment of the teacher j to a particular timeslot n. The aim of this variable is to control the costs of the model in a way that no teacher is paid more than once for one timeslot, which could be the case when more than one solo-flying student is assigned to a teacher in one timeslot. Thus, $d_n^j = 1$ if the teacher j is assigned to one or more student i in a timeslot n, else $d_n^j = 0$.

4.2 PROBLEM FORMULATION

This section presents a mathematical formulation of the problem. The mathematical model includes the mix between classical formulations used in the literature for course timetabling problems, as well as specific constructions that are requested for flight school course timetabling problems. Furthermore, the course timetabling problem was chosen as a base for the model created in this thesis due to its resemblance with the problem requirements. The formulations that are used for the model were taken from a number of papers, presented and discussed in a detail in Chapter 2.2.

First of all, the objective function should be formulated. As mentioned before, most of the course timetabling problems set an objective function, which either

minimizes the penalties for hard constraint violations or maximizes the preferences of students or teachers. As the model that is elaborated in this thesis aims for the best feasible timetable, which could respect all requirements, and thus does not have a goal to achieve a best runtime, the second alternative for the creation of the objective function is used. Namely, the model aims to maximize the matching between the students' monthly amount of available/ preferred timeslots with actually scheduled timeslots or in other words it maximizes the number of assigned lectures (Schaerf, 1999; Lach & Lübbecke, 2012):

$$\max \sum_{n=1}^{N} \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{z=1}^{Z} x_{njz}^{i}$$

However, this objective function neglects the cost difference between external and internal instructors, as well as the variance of different plane type expenses. In order to differentiate between the types of costs, distinctive values for internal and external teachers are introduced. To prevent instructors from receiving triple payment for one timeslot in case when more than one student is assigned to instructor j in timeslot n, a distinctive decision variable d_n^j must be added to the objective function. Variable $d_n^j = 1$ if and only if a teacher j is assigned to timeslot n, regardless of how many simultaneous lectures he holds in this hour. In this way, teacher's payment for hour is calculated only once and thus forces the model to assign more students to one teacher in one timeslot so as to maximize profit.

Moreover, different costs values for planes are added as well. The costs are then reduced from profit value. The use of profit/cost differentiation function not only makes objective function more detailed and applicable in practice but also makes it more optimized and faster to solve because of the shorter process of matching cases. Therefore, in this way the problem is relaxed.

The complete objective function (1) is thus formulated as:

$$\max \sum_{n=1}^{N} \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{z=1}^{Z} x_{njz}^{i} \left(P - C_{PLANE_z}\right) - \sum_{j=1}^{J} \sum_{n=1}^{N} d_n^j * C_{INSTRUCTOR_j} \qquad (1)$$

Furthermore, the number of constraints has to be enforced. The first two hard constraints represent the basic requirements for course timetabling problems and base on the formulations of Schaerf (1999). Namely, constraints (2) ensure that each

student attends or is assigned only to one class, one teacher and one plane for each timeslot. Constraints (3) ensure that each plane is assigned only to one student and one teacher for each timeslot.

$$\sum_{z=1}^{Z}\sum_{j=1}^{J} x_{njz}^{i} \leq 1 \qquad \forall i \in I;\ \forall n \in N \tag{2}$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} x_{njz}^{i} \leq 1 \qquad \forall z \in Z;\ \forall n \in N \tag{3}$$

The third basic constraint should guarantee that a teacher is not involved into more than one lecture for each period. Literature proposes (cf. Schaerf, 1999; McCollum et al., 2007; Lach & Lübbecke, 2012) to use a following constraint for ensuring this in course timetabling problem:

$$\sum_{i=1}^{I}\sum_{z=1}^{Z} x_{njz}^{i} \leq 1 \qquad \forall j \in J;\ \forall n \in N$$

However, particularly for the problem formulated in this thesis, an additional detail concerning the flight school timetable should be added to the constraint. Explicitly, it should be distinguished whether a teacher is teaching in the air, while flying with only one student, or from the ground, while observing maximum three students at the same time. Constraints (4) are thus formulated as following:

$$\sum_{i=1}^{I}\sum_{z=1}^{Z} b_{n}^{i}\, x_{njz}^{i} \leq 1 \qquad \forall j \in J;\ \forall n \in N \tag{4}$$

$$where\ b_{n}^{i} \in \{\frac{1}{3}\ for\ solo\ flights\ (j\ on\ the\ ground); 1\ for\ dual\ flights\ (j\ in\ the\ air)\}$$

Constraints (4) are non-linear, thus a transformation to a linear formulation is needed. This is ensured through a mathematical expression that uses an auxiliary variable. Constraints (5) force variable y_{njz}^{i} to take the value of a non-linear product of two multipliers, b_{n}^{i} and x_{njz}^{i}. Therefore, the constraint (4) can be replaced with the constraint (5), which expresses the same value formulation as constraint (4) but in a non-linear construction:

$$\begin{cases} y_{njz}^{i} \leq x_{njz}^{i} \\ y_{njz}^{i} \leq b_{n}^{i} \\ y_{njz}^{i} \geq b_{n}^{i} - (1 - x_{njz}^{i}) \end{cases} \qquad \forall i \in I;\ \forall j \in J; \forall z \in Z;\ \forall n \in N \tag{5}$$

Therefore, the constraint (4) can be replaced with the constraint (4a), which expresses the same value formulation as constraint (4) but in a non-linear construction and thus is similar to the basic constraints proposed in the course timetabling literature (Alvarez-Valdes et al., 2002; Burke et al., 2009; MirHassani & Habibi, 2013):

$$\sum_{i=1}^{I}\sum_{z=1}^{Z} y_{njz}^{i} \leq 1 \qquad \forall j \in J;\ \forall n \in N \tag{4a}$$

Next important constraint (6) ensures that only available timeslots are considered. Therefore, the student cannot be assigned to a timeslot, which he has not indicated as preferable. In this way, constraint (6) makes a connection between input and output of the model.

$$x_{njz}^{i} \leq AVAL_{n}^{i} \qquad \forall i \in I;\ \forall z \in Z;\ \forall j \in J;\ \forall n \in N \tag{6}$$

The constraint (7) guarantees the maximum amount of school opening hours per month. Thus constraint (7) ensures that the number of assigned timeslots does not overweight the total amount of working hours of all instructors or the maximum amount of monthly available timeslots. In addition, constraint (7a) limits the number of working hours only to those in which the instructor was actually assigned to one or more students. In this way, constraints (7) and (7a) prohibit each teacher of being overpaid.

$$\sum_{n=1}^{N} d_{n}^{j} \leq MAX\_N \qquad \forall j \in J \tag{7}$$

$$If \sum_{i=1}^{I}\sum_{z=1}^{Z} x_{njz}^{i} > 0\ then\ d_{n}^{j} = 1\ else\ d_{n}^{j} = 0 \qquad \forall j \in J, n \in N \tag{7a}$$

As to the numerical values for the total amount of timeslots, those can be easily varied in the model. For example, an Austrian company may act in accordance with Austrian labour authorities, which regulate that the full-time employment consists from 40 hours/week. Part-time employment impels less than 40 working hours/week, whereby usually consists from 20hours/week. (WKO, 2014)

Constraint (8) sets the maximum amount of consecutive flying hours (= block time) per day (noted as r), whereas a duration of the working day is assumed to be 12

hours. It also ensures that the last few hours of a day and the first few hours of the following day are not considered as consecutive hours.

$$\sum_{z=1}^{Z}\sum_{j=1}^{J}\sum_{q=1+12r}^{12+12r} x_{qjz}^{i} \leq S \quad \forall i \in I;\ \forall r\ \in [0 \dots 20] \tag{8}$$

Further, the calculation of completed hours (β_n^i) should be enforced in order to include the information about current course characteristics. Constraint (9) counts the amount of competed hours by student i until the current hour α:

$$\beta_n^i = \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{\alpha=1}^{n} x_{\alpha jz}^{i} + PH^i \qquad \forall i \in I; \forall n \in N \tag{9}$$

$$where\ PH^i = \beta_N^i\ from\ the\ previous\ planning\ period \qquad \forall i \in I$$

In addition, it should be ensured (constraint (10)) that a student is not assigned to any further courses if he has completed a full program. For instance, according to the verbal model requirements of this thesis, the ATPL course is completed then the value for β_n^i reaches 141 completed hours by a student i. Thus, the parameter FULL_P should be set to 142 hours in order to indicate that the student i has completed all requiring flying hours and should not participate in any further lectures.

$$\sum_{j=1}^{J}\sum_{z=1}^{Z} x_{njz}^{i} \leq FULL_P - \beta_n^i \qquad \forall i\ \in I;\ \forall n\ \in N \tag{10}$$

Constraint (11) assigns a proper plane to the course, according to the number of already completed hours by a student i. Obeying requirements set by the school (see Table 3-2), a model should forbid the use of plane type 1 and 2 (i.e. C142S and C142R) in the last course, whereby exactly in this course plane type 3 should be assigned. Keeping in mind that the last course is identical to $\beta_n^i = [135; 141]$, the logical expression should be formulated as follows:

$$If\ \beta_n^i \in [135 \dots 141]\ then \sum_{j=1}^{J}(x_{nj1}^{i} + x_{nj2}^{i}) = 0\ \ else \sum_{j=1}^{J} x_{nj3}^{i} = 0\ \ \forall i\ \in I;\ \forall n\ \in N \tag{11}$$

Similarly, according to the value of β_n^i, a decision can be made whether a student i should be engaged in a solo or a dual flight course. Regarding the case-based requirements, a student i is flying alone in the courses K3 and K5, which correspond to the $\beta_n^i = [21; 30]$ and $\beta_n^i = [36; 75]$ respectfully (see Table 3-2).

Therefore, constraint (12) forces a variable b_n^i to take correct values depending on the β_n^i:

$$b_n^i = \begin{cases} \frac{1}{3} & if\ \beta_n^i \in [21 \dots 30] \cup [36 \dots 75] \\ 1 & otherwise \end{cases} \qquad \forall i\ \in I;\ \forall n\ \in N \qquad (12)$$

The last two constraints also depend on β_n^i and forbid a lecturer from teaching courses, which do not correspond to his teaching license (constraint (13)), as well as set evening time for the night courses (constraint (14)).

$$If\ \beta_n^i\ \geq 81\ then \sum_{z=1}^{Z} x^i_{n,\text{"Mr. Beginner"},z} = 0\ \ else \sum_{j=1}^{J} x^i_{njz} \leq 1\ \ \forall i\ \in I;\ \forall n\ \in N \qquad (13)$$

(14)

$$If\ \beta_n^i\ \epsilon\ [76; 80]\ then \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{q=12r+1}^{12r+9} x^i_{qjz} = 0\ else \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{q=12r+1}^{12r+9} x^i_{qjz} \leq 1\ \ \forall i \in I; \forall r \in\ [0 \dots 20]$$

Finally, constraint (15) proves the binary characteristics of the decision variable x_{njz}^i. In addition, it should be mentioned that all variables are non-negative.

$$x_{njz}^i \in [0,1] \qquad \forall i \in I;\ \forall n\ \in N;\ \forall j\ \in J; \forall z\ \in Z \qquad (15)$$

The constraints listed above describe all of the pre-given verbal requirements of the problem (see Chapter 3.2) and thus can now be modelled in the solver in order to achieve a practical solution. In addition, a number of further (case-based) limitations or needed requirements can be set as logical expressions in a connection with the variable β_n^i. A complete list of the problem's objective function and constraints is given on the next page for the better overview purposes. The constraints, which were neglected in simplified model are marked with a "star" * (more details in Chapter 5.2).

$$\max \sum_{n=1}^{N}\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{z=1}^{Z} x_{njz}^{i}\left(P - C_{PLANE_z}\right) - \sum_{j=1}^{J}\sum_{n=1}^{N} d_n^j * C_{INSTRUCTOR_j} \quad (1)$$

$$\sum_{z=1}^{Z}\sum_{j=1}^{J} x_{njz}^{i} \le 1 \quad \forall i \in I;\ \forall n \in N \quad (2)$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} x_{njz}^{i} \le 1 \quad \forall z \in Z;\ \forall n \in N \quad (3)$$

$$\sum_{i=1}^{I}\sum_{z=1}^{Z} b_n^i\, x_{njz}^{i} \le 1 \quad \forall j \in J;\ \forall n \in N \quad (4*)$$

$$\sum_{i=1}^{I}\sum_{z=1}^{Z} y_{njz}^{i} \le 1 \quad \forall j \in J;\ \forall n \in N \quad (4a*)$$

$$\begin{cases} y_{njz}^{i} \le x_{njz}^{i} \\ y_{njz}^{i} \le b_n^i \\ y_{njz}^{i} \ge b_n^i - (1 - x_{njz}^{i}) \end{cases} \quad \forall i \in I;\ \forall j \in J; \forall z \in Z;\ \forall n \in N \quad (5*)$$

$$x_{njz}^{i} \le AVAL_n^i \quad \forall i \in I;\ \forall z \in Z;\ \forall j \in J;\ \forall n \in N \quad (6)$$

$$\sum_{n=1}^{N} d_n^j \le MAX\_N \quad \forall j \in J \quad (7)$$

$$If \sum_{i=1}^{I}\sum_{z=1}^{Z} x_{njz}^{i} > 0\ then\ d_n^j = 1\ else\ d_n^j = 0 \quad \forall j \in J;\ \forall n \in N \quad (7a)$$

$$\sum_{z=1}^{Z}\sum_{j=1}^{J}\sum_{q=1+12r}^{12+12r} x_{qjz}^{i} \le S \quad \forall i \in I;\ \forall r \in [0 \dots 20] \quad (8)$$

$$\beta_n^i = \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{\alpha=1}^{n} x_{\alpha jz}^{i} + PH^i \quad \forall i \in I;\ \forall n \in N \quad (9)$$

$$\sum_{j=1}^{J}\sum_{z=1}^{Z} x_{njz}^{i} \le FULL_P - \beta_n^i \quad \forall i \in I;\ \forall n \in N \quad (10)$$

$$If\ \beta_n^i \in [135 \dots 141]\ then \sum_{j=1}^{J}(x_{nj1}^{i} + x_{nj2}^{i}) = 0\ else \sum_{j=1}^{J} x_{nj3}^{i} = 0 \quad \forall i \in I;\ \forall n \in N \quad (11)$$

$$b_n^i = \begin{cases} \frac{1}{3} & if\ \beta_n^i \in [21 \dots 30] \cup [36 \dots 75] \\ 1 & otherwise \end{cases} \quad \forall i \in I;\ \forall n \in N \quad (12*)$$

$$If\ \beta_n^i \ge 81\ then \sum_{z=1}^{Z} x_{n,\text{"Mr. Beginner"},z}^{i} = 0\ else \sum_{j=1}^{J} x_{njz}^{i} \le \quad \forall i \in I;\ \forall n \in N \quad (13)$$

(14)

$$If\ \beta_n^i\ \epsilon\ [76; 80]\ then \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{q=12r+1}^{12r+9} x_{qjz}^{i} = 0\ else \sum_{j=1}^{J}\sum_{z=1}^{Z}\sum_{q=12r+1}^{12r+9} x_{qjz}^{i} \le 1 \quad \forall i \in I;\ \forall r \in [0 \dots 20]$$

$$x_{njz}^{i} \in [0,1] \quad \forall i \in I;\ \forall n \in N;\ \forall j \in J; \forall z \in Z \quad (15)$$

5. COMPUTATIONAL EXPERIMENTS

This chapter presents computational experiments that were done using real test data in order to find out whether the mathematical model illustrated in this thesis (Chapter 4) is applicable to the practical issues. The problem was formulated as MIP in Fico Xpress Optimization Suite (Version 7.6 64bit) using Mosel programming language. Mosel language is a tool for modelling and solving problems, as it offers a language that can be used both for modelling and programming. Mosel also provides different solution algorithms for each problem with the aim to guarantee higher degree of efficiency by complex models. (Dash Associates, 2003)

The calculations were implemented on the Intel ® Core ™ i5-3317U CPU @ 1.70 GHz, 4 GB RAM on Windows 8.1 64bit. In sum, three data instances were tested, which are described in detail further in this chapter (Ch. 5.1). Afterwards, the chapter 5.2 continues with presenting experimental results of the programming. Finally, chapter 5.3 proposes some managerial implications and discusses the applicability of the model to the reality.

5.1 DATA INSTANCES

This thesis used three different scenarios with diverse sets of data (Table 5-1). All three data instances used the same amount of teachers and planes, which corresponds to a real situation of the flight school. Namely, by the time of data collection, school had three practice airplanes (i.e. C142S, C142R, DA42), three full-time teachers and several part-time instructors with random availabilities. As the number and choice of the part-time teachers is always varying, this paper assumes only one part-time instructor, who is always available.

Table 5-1: Details of the experimental data sets

Name	Students	Teachers	Planes	Timeslots/days
Instance_test	3	4	3	12/1
Instance_BIG	10	4	3	60/5
Instance_REAL	20	4	3	240/20

For the calculation purposes, the revenue and cost values are non-realistic but are rather set as constants. Nonetheless, these constants are chosen in a way to

differentiate between different price levels of planes and teachers in order to reflect the reality as close as possible. Following values for profit parameters were used in all three instances:

- P = 5
- $C\_INSTRUCTOR_j$: full = 1; part = 2
- C_PLANE_z: C142 = 1; DA42 = 2

In addition to the similarities of the data instances mentioned above, the sets were distinguished in the number of students and timeslots. The goal of the first and the smallest set (Instance_test) was to test whether the processing of the mathematical model into a programming code was successful. Therefore, it represented a very simplified version of the real data in order to achieve results in a minimum running time.

The second instance (Instance_BIG) concerned 10 students and 60 timeslots, thus calculated the timetable for one working week consisting of 5 days. Although, this set was self-generated, it reflects the reality quite close as in less active months the number of students in the school does not exceed 10. Yet, in this case the problem should be solved each week in comparison to the requested one-month planning period.

The third instance (Instance_REAL) was adapted to the real data available at school at the time of the data collection. Namely, school had 20 students in several different learning phases and hence with varying value of previous hours (parameter PH^i). In addition, this instance considered monthly planning period by generating 240 timeslots or 20 working days.

The availabilities of the students were randomly generated in MS Excel, whereas past real data was partly set as a base for generation.

5.2 EXPERIMENTAL RESULTS

As already mentioned, the problem was formulated as MIP and solved in Xpress software, which tried to find a solution by applying an exact algorithm, namely Branch-and-Bound. For this goal the mathematical problem was reformulated into a Mosel language. A full code of the program can be found in Appendix A.

Table 5-2 shows a comparison of obtained solutions and runtimes for different sets of the model. The column “Time” shows runtime for each set. After a number of

computational experiments, the maximum Xpress running time of the problem was limited to 4000 seconds, where most of the optimal solutions were found. Experiments in scope of this thesis showed that longer running times had no influence on the significantly better results but rather brought single accidental solutions or just slightly decreased the "Gap".

Table 5-2: Computational results in Xpress (exact method)

Instance	**Solution**	**Time (s)**	**Gap**	**# of feasible solutions**
In_test	100	0.3	0.00%	Optimal
In_BIG	407	4000	4,03%	15
In_BIG_simple	386	4.8	0.00%	Optimal
In_REAL	n/a	>40000	1e+042%	n/a
In_REAL_simple	1600	1262.4	0.00%	Optimal

First of all, the smallest instance was tested in order to test the performance of the model. Apparently, Xpress did not have any difficulties with finding a solution for the small test set (Figure 5-1). The code has been proven for working successfully. Yet, in practice, it is hardly realistic and highly unprofitable to create a timetable for one-day period only.

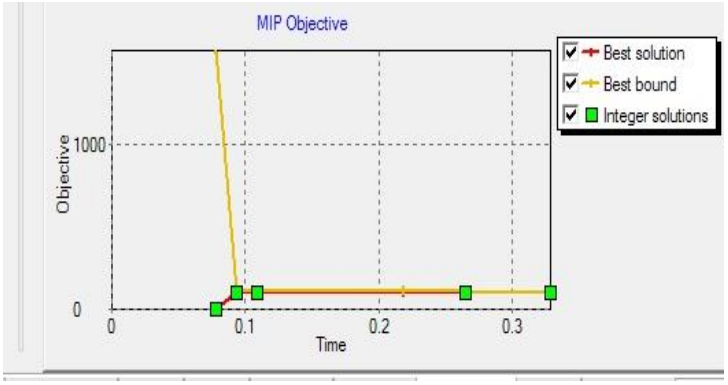


Figure 5-1: Solutions for Instance_test

After testing the code, the solution for Instance_BIG was searched. An increased number of constraints of the bigger problem have brought significant difficulties for the Xpress. After 4000s of running time, a problem still had a "Gap" of 4.03% and an optimal solution was not found. Up to this time Xpress has found 15 integer

solutions, which created 15 feasible timetables (Figure 5-2). An example of a feasible timetable for a 5-day problem is demonstrated in Appendix B.

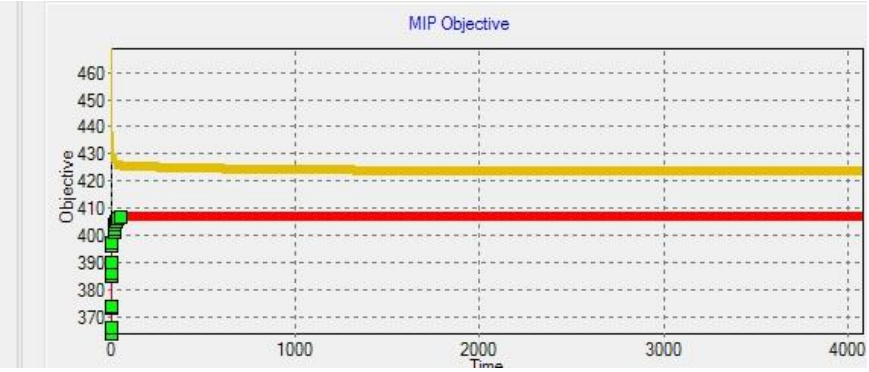


Figure 5-2: Solutions for Instance_BIG after 4000s running time

The similar situation also occurred, while running the Instance_REAL set. In this case, Xpress had to struggle with even larger amount of constraints and did not find an optimal solution neither in 4000s limit runtime or longer experiments with over 40 000s running time. Even after 12 hours of running time Xpress did not come with any solutions, had enormous "Gap" and was still searching for the first integer solution.

After running several unsatisfying experiments with a complete model (Instance_REAL and Instance_BIG both full versions), a decision was made to simplify the model by removing several soft constraints. As discussed in Chapter 2, a removal of soft constraints decreases the quality of the timetable, yet at the same time it improves the probability of finding an optimal solution, as well as an algorithm speed. It was decided to remove an option of an instructor to teach several students at once. Therefore, a decision variable b_n^i that differentiated between solo and dual flights was removed. In a connection with the removed decision variable, constraints that are marked with the star in a complete model overview in Chapter 4.2 (i.e. 4, 4a 5, and 12) were neglected as well. After the alternation, the model has gained the common three hard constraints as proposed in the literature (see Chapter 2.2). Moreover, the quality of the timetable in practical terms was not worsened significantly. In reality, teachers are mostly flying with the students and hence observe three students at once quite rarely.

A simplified problem was run for Instance_BIG and Instance_REAL (Figure 5-3). Both sets produced optimal solution within very short and short runtime accordingly (Table 5-2). Simplified model for the set Instance_BIG_simpl had found

in 4.8s an optimal solution equal to 386, which is just 5.16% worse than the best solution found in 4000s by a full model. Relaxed model for the biggest set (Instance_REAL_simpl) was found in 21 minutes and produced an optimal solution of 1600, which is significantly better than the previous run of the full model, which did not produce any feasible results neither in 4000s runtime limit nor after 12 hours of calculations.

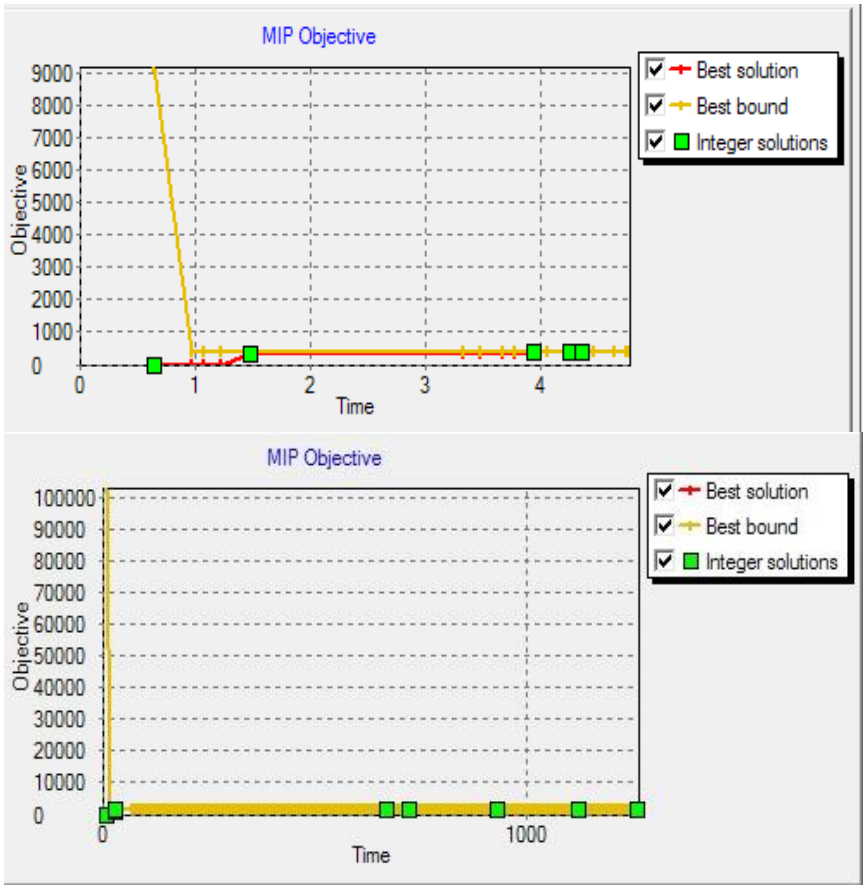


Figure 5-3: Solutions for simplified models (Instance_BIG_simpl and Instance_REAL_simpl)

5.3 MANAGERIAL IMPLICATIONS

The computational results gave useful insights into the applicability of the solution method for course timetable problems to the reality. It was proven that the method described in this paper can be applied to solve real timetabling problems of the small-to-medium sized educational enterprises. Before using the model, strategic managers or planners should evaluate their individual goals and hence the requirements of the real-problem.

There are two options to be considered in this case:

(1) Use simplified model with larger amount of soft constraints that can be partly or fully violated;
(2) Plan and create short-term timetables.

Option (1) produces feasible results in a short time. However, it has a negative influence on the quality of the timetable. Therefore, planners should decide thoughtfully, which soft constraints can be actually violated in order that the timetable would still remain close to the reality and represent managerial requirements.

Option (2) considers zero violations of the constraints, hence produces realistic timetables. For instance, according to the calculations done in scope of this thesis, the set “Instance_BIG” has produced the most optimal results in a realistic runtime, despite its short planning period. It implies that the timetable planner should consider the opportunity to create short one-week timetables if the aim of the planning strategy is to generate optimal MIP timetables in a very quick running time. Yet, timetables with one-week duration are in fact the largest disadvantage of the second option, as are quite impractical.

6. CONCLUSION

This thesis developed and described a mixed integer program for the assignment of air fleet and teaching stuff to students of a flight school. The objective of this program was to maximize the amount of held lectures in a month period under the consideration of the company's profit gains from each class. The model was partly based on the university course timetabling problem, which belongs to the timetabling family together with school and university examination timetabling.

The problem was tested with three instances of different size and solved with Xpress solver. First, through solving smaller instances, the mathematical model was proven to be well applicable to the real timetable problems. Nonetheless, the solver was able to produce good results only for the smallest instance. Solving large instances required impractically long computational times. Secondly, a simplified model that violated several constraints was applied to test larger instances. In this case solver had produced feasible results within short runtime.

Nevertheless, long computational times were not considered as a significant disadvantage of the model as the aim of this paper was to formulate a real-case course timetable problem as an integer program and then to find a practical solution. The goal of the method was to find any feasible solution (i.e. create a usable timetable) and not the best possible solution in a benchmark time (cf. ITC 2011). In fact, researchers also identify the primary use of the integer programming as "[…] a powerful modelling tool for formal and precise identification of relevant decisions and constraints […] than (*as a tool for, auth.*) actually trying to compute optimal solutions with its help" (Lach & Lübbecke, 2012, p. 269). However, Lach & Lübbecke (2012), MirHassani & Habibi (2013) and some other researchers (see Chapter 2.2) believe that a big progress has been recently made in IP solver technologies that stimulate a use of non-standard algorithms and computations with advanced solvers.

Therefore, it would be interesting to solve the flight school problem with the help of diverse metaheuristics to achieve better and faster solutions. Besides, use of another more powerful solver than Xpress may be practical to compute large problems.

Furthermore, the author of this paper did not use any particular parameter tuning to the instances and the problem in general. Thus, some advanced tuning and optimization could be useful in practical terms of the problem. Nonetheless, the application of algorithms and parameter tuning lies beyond the scope of this paper.

Finally, as already mentioned above, the approach described in this paper has proven to create feasible timetables for university course timetable problems. However, further improvements may be applied to the model. One of the most important further considerations includes introducing penalties for assigning different teachers and/or planes for a particular student during a number of consecutive hours. A chance not to change teachers/planes during a long lecture, consisting of several hours/timeslots, would make the timetable significantly more comfortable for practical use, as well as increase its compactness.

BIBLIOGRAPHY

Abramson, D., & Abela, J. (1992). A Parallel Genetic Algorithm for Solving the School Timetabling Problem. *15 Australian Computer Science Conference*, (pp. 1-11). Hobart.

Abramson, D., Krishnamoorthy, H., & Dang, H. (1996). Simulated annealing cooling schedules for the school timetabling problem. *Asia-Pasific Journal of Operational Research*, 1-22.

Alvarez-Valdes, R., Crespo, E., & Tamarit, J. (2002). Design and implementation of a course scheduling system using Tabu Search. *European Journal of Operational Research*, 512-523.

Bateburg, K., & Palenstijn, W. (2003). A new exam timetabling algorithm. *Belgian-Dutch Artificial Intelligence Conference (BNAIC)*, (pp. 19-26).

Brito, S., Fonseca, G., Toffolo, T., Santos, H., & Souza, M. J. (2012). A SA-VNS approach for the High School Timetabling Problem. *Electronic Notes in Discrete Mathematics*, 169-176.

Burke, E., Marecek, J., Parkes, A., & Rudova, H. (2009). Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 582-597.

Cambazard, H., Demazeau, F., Jussien, N., & David, P. (2004). Interactively solving school timetabling problems using extensions of constraint programming. *5th International Conference on the Practice and Theory of Automated Timetabling*, (pp. 107-124). France.

Cambazard, H., Herbrard, E., O'Sullivan, B., & Papadopoulos, A. (2012). Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals of Operations Research*, 111-135.

Carrasco, M., & Pato, M. (2001). A multiobjective genetic algorithm for the class/teacher timetabling problem. In E. Burke, & W. Erben, *Practice and Theory of Automated Timetabling (PATAT) III* (pp. 3-17). Berlin: Springer-Verlag.

Carter, M. (1989). A Lagrangian Relaxation approach to classroom assignment problem. *INFOR*, 230-246.

Carter, M. (2001). Timetabling. In S. Gass, & C. Harris, *Encyclopedia of Operations Research and Management Science* (pp. 833-836). Kluwer Academic Publishers.

Carter, M., & Laporte, G. (1997). Recent developments in practical course timetabling. In E. Burke, & M. Carter, *Practice and Theory of Automated Timetabling II* (pp. 3-19). Berlin: Springer-Verlag.

Dash Associates. (2003). *Xpress-Mosel Language Reference Manual.* New Jersey: Dash Optimization Inc. Retrieved from http://home.deib.polimi.it/malucell/didattica/appunti/mosel/mosel-language_reference.pdf

Daskalaki, S., & Birbas, T. (2005). Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 106-120.

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 117-135.

De Causmaecker, P., Demeester, P., & Berghe, G. (2009). A decomposed metaheuristics approach for a real-world university timetabling problem. *European Journal of Operational Research,* 307-318.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research,* 151-162.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. M., & Soumis, F. (1997). Daily Aircraft Routing and Scheduling. *Management Science,* 841-855.

Desrochers, M., & Soumis, F. (1989). A column generation approach to the urban transit Crew Scheduling Problem. *Transportation Science,* 1-13.

Di Gaspero, L., & Schaerf, A. (2002). Multi-neighbourhood local search with application to course timetabling. In E. Burke, & P. De Causmaecker,

Practice and Theory of Automated Timetabling (PATAT) (pp. 263-287). Berlin: Springer-Verlag.

Di Gaspero, L., McCollum, B., & Schaerf, A. (2007). *Curriculum-based Course Timetabling (Track 3). Technical Report.* The Second International Timetabling Competition (ITC- 2007).

Dimopoulou, M., & Miliotis, P. (2001). Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, 202-213.

Dinkel, J., Mote, J., & Venkataramanan, M. (1989). An efficient decision support system for academic course scheduling. *Operations Research*, 853-864.

Eikelder, H., & Willemen, R. (2001). Some Complexity Aspects of Secondary School Timetabling Problems. In E. Burke, & W. Erben, *Practice and Theory of Automated Timetabling III.* (pp. 18-27). Konstanz, Germany: Springer-Verlag Berlin Heidelberg.

Elmohamed, S., Fox, G., & Coddington, P. (1998). A comparison of annealing techniques for academic course scheduling. In E. Burke, & M. Carter, *Practice and Theory of Automated Timetabling (PATAT)* (pp. 146-166). Berlin: Springer-Verlag.

Even, S., Itai, A., & Shamir, A. (1976). On the complexity of timetabling and multicommodity flow problems. *SIAM Journal of Computation*, 691-703.

Fahrion, R., & Dollansky, G. (1992). Construction of university faculty timetables using logic programming technique. *Discrete Applied Mathematics*, 221-326.

Gamache, M., Sounis, F., Marquis, G., & Desrosiers, J. (1994). *A column generation approach for large scale aircrew rostering problems.* Montrela, Canada: Ecole des Hautes Etudes Commericales.

Gogos, C., Alefragis, P., & Housos, E. (2012). An improved multi-stages algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 203-221.

Gotlieb, C. C. (1963). The construction of class-teacher timetables. *IFIP Congress 62* (pp. 73-77). North-Holland: C.M. Popplewell.

Hahn-Goldberg, S. (2007). *Defining, Modelling, and Solving a Real University Course Timetabling Problem.* Toronto: University of Toronto.

Hertz, A. (1992). Finding a feasible course scheduling using tabu search. *Discrete Applied Mathematics*, 255-270.

J., J., & Lassez, J. (1987). Constraint Logic Programming. *POPL 87 on Principles of Programming Languages* (pp. 111-119). New York: ACM.

JA Flight Training. (2013). *Ausbildungsprogram. Booklet.* Kottingbrunn, Austria.

Jaziri, W. (2008). *Local Search Techniques: Focus on Tabu Search.* Croatia: In-Tech.

Junginger, W. (1986). Timetabling in Germany - a survey. *Interfaces*, 306-317.

Kohl, N. (2003). *Solving the world's largest crew scheduling problem.* Retrieved September 15, 2013, from Citeseer: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.1044

Lach, G., & Lübbecke, M. (2012). Curriculum based course timetabling: new solutions to Udine benchmark instances. *Annals of Operational Research*, 255-272.

Laporte, G., & Desroches, S. (1986). The problem of assigning students to course sections in a large engineering school. *Computers and Operations Research*, 387-394.

Lettovsky, L., Johnson, E., & Nemhauser, G. (2000). Airline crew recovery. *Transportation Science*, 337-348.

Lewis, R. (2007). *A survey of metaheuristics-based techniques for university timetabling problems.* Cardiff: Cardiff Business School, Pryfysgol Caerdydd/Cardiff University.

Lu, Z., & Hao, J. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 235-244.

Malik, A., Ayob, M., & Hamdan, A. (2010). Stratified random sampling technique for integrated two-stage mutli-neighbourhood tabu search for examination

timetabling problem. *Intelligent Systems Design and Applications*, 1326-1331.

McCollum, B., McMullan, P., Burke, E., Parkes, A., & Qu, R. (2007). *The Second International Timetabling Competition: Examination Timetabling Track.* UK: School of Computer Science.

Metzenbauer, M. (2013, November 29). *Ende für JA Flight Training*. Retrieved March 24, 2014, from Austrian Aviation: http://www.austrianaviation.net/news-regional/news-detail/datum/2013/11/29/ende-fuer-ja-flight-training.html

Metzenbauer, M. (2014, March 3). *Rettung für Jetalliance-Simulator*. Retrieved March 24, 2014, from Austrian Aviation: http://www.austrianaviation.net/news-regional/news-detail/datum/2014/03/03/rettung-fuer-jetalliance-simulator.html

MirHassani, S., & Habibi, F. (2013). Solution approaches to the course timetabling problem. *Artificial Intelligence Review*, 133-149.

Miyaji, I., Ohno, K., & Mine, H. (1987). Solution method for partitioning students into groups. *European Journal of Operational Research*, 82-90.

Petrovic, S., & Burke, E. (2004). *University timetabling.* Nottingham: School of Computer Science and Information Technology, University of Nottingham.

Pillay, N., & Banzhaf, W. (2009). A study of heuristic combinations for hyper-heuristic systems for the uncapacited examination timetabling problem. *European Journal of Operational Research*, 482-491.

Pongcharoen, P., Promtet, W., Yenradee, P., & Hicks, C. (2008). Stochastic optimisation timetabling tool for university course scheduling. *International Journal of Production Economics*, 903-918.

Post, G., Ahmadi, S., & Geertsema, F. (2012). Cyclic transfers in school timetabling. *OR Spectrum*, 133-154.

Qaurooni, D., Lanzi, P., & Krasnogor, N. (2011). A memetic algorithm for course timetabling. *Genetic and evolutionary computation conference* (pp. 435-442). Tehran, Iran: Association for Computing Machinery.

Qu, R., Burke, E., McCollum, B., & Merlot, L. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 55-89.

Raghavjee, R., & Pillay, N. (2012). A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. *2012 Fourth World Congress on Nature and Biologically Inspired Computing* (pp. 98-103). IEEE Publishing.

Ross, P., Hart, E., & Corne, D. (2003). Genetic algorithms and timetabling. *Natural computing series, advances in evolutionary computing: theory and applications*, 755-77.

Schaerf, A. (1996). Tabu search techniques for large high-school timetabling problems. *Proc. of the 13th National Conference on Artificial Intelligence* (pp. 363-368). Portland, USA: AAAI Press/MIT Press.

Schaerf, A. (1999). *A Survey of Automated Timetabling.* Rome: Dipartimento di informatica e Sistemistica. Universita di Roma "La Sapienza".

Schmidt, G., & Strohlein, T. (1979). Timetable construction - an annotated bibliography. *The Computer Journal*, 307-316.

Sherali, H. D., Bish, E., & Zhu, X. (2006). Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 1-30.

Socha, K., Sampels, M., & Manfrin, M. (2003). Ant algorithm for the university course timetabling problem with regard to the state-of-the-art. *EvoCOP*, (pp. 334-345). Essex, UK.

Stojkovic, M., Soumis, F., & Desrosiers, J. (1998). The operational airline crew scheduling problem. *Transportation Science*, 252-245.

Tassapoulos, I., & Beligiannis, G. (2012). A hybrid particle swarm optimization based algorithm for high school timetabling problems. *Applied Soft Computing*, 3472-3489.

Tripathy, A. (1992). Computerised decisions aid for timetabling - A case analysis. *Discrete Applied Mathematics*, 313-323.

Vance, P., Atamtürk, A., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., . . . Rebello, R. (1997). *A heuristic Branch-and-Price approach for the airline Crew Pairing Problem.* Retrieved September 15, 2013, from http://www.dim.uchile.cl/~tcapelle/TESIS/papersGeneracionColumnas/lec9706.pdf

WKO. (2014, February 14). *Arbeitszeit*. Retrieved March 20, 2014, from WKO. Wirtschaftskammer Österreich: https://www.wko.at/Content.Node/Service/Arbeitsrecht-und-Sozialrecht/Arbeitsrecht/Arbeitszeit/Arbeitszeit---Themenstartseite.html

Wren, A. (1996). Scheduling, Timetabling and Rostering - A Special Relationship? In E. Burke, & P. Ross, *The Practice and Theory of Automated Timetabling: Selected Papers from 1st International Conference on the Practice and Theory Automated Timetabling* (pp. 46-75). Napier University.

Yoshikawa, M., Kaneko, K., Nomura, Y., & Watanabe, M. (1996). A constraint-based high school scheduling system. *IEEE Expert*, 63-72.

Zeghal, F., & Minoux, M. (2006). Modelling and solving a Crew Assignment Problem in air transportation. *European Journal of Operational Research*, 187-209.

Zhang, D., Liu, Y., M'Hallah, R., & Leung, S. (2010). A simulated annealing with a new neighbourhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 550-558.

APPENDIX A: PROGRAM CODE

```
model "JA_AP"
uses "advmod"; !gain access to the Xpress-Optimizer solver
declarations
! dimensions
    NN = 240
    N = 1..NN              ! time slots
    J: set of string       !teachers
    I: set of integer      ! students
    Z = {"C142S", "C142R", "DA42"}   ! planes
    Q = 1..11              !courses

! data and variables
    AVAL: array(I,N) of integer          ! Availabilities of the students
    P: real                              ! Revenue per student
    S: integer                           ! Max flying hours per day
    C_PLANE: array(Z) of integer         ! Costs for plane
    C_INSTRUCTOR: array(J) of integer !Costs for instructors
    PH: array(I) of integer              ! Previously completed hours
    q_course: dynamic array(I,N) of integer   ! Course number
    c: array (range) of linctr           ! Auxiliary variable
    y: array (I,N,J,Z) of mpvar          ! Product of b and x
    b: array(I,N) of mpvar               ! If instructor is on ground or in air
    d: array (J,N) of mpvar              ! Assignment of a teacher
    x: dynamic array(I,N,J,Z) of mpvar   ! If a student is assigned to a timeslot
    beta: dynamic array(I,N) of mpvar    ! Number of completed hours
  ! access to data file
    FILENAME = "JAAP_data_REAL.dat"
  end-declarations

  initialisations from FILENAME
    N I J AVAL C_PLANE P C_INSTRUCTOR PH S
  end-initializations
  finalize(N); finalize(I)
  forall(i in I, n in N, j in J, z in Z)
    create(x(i,n,j,z))
  forall(i in I, n in N) do
    create(beta(i,n))
    end-do
```

```
! 1. objective function
Y := sum(i in I, n in N, j in J, z in Z)
 x(i,n,j,z) *(P - C_PLANE(z)) – (sum(n in N, j in J) d(j,n)*C_INSTRUCTOR(j))

!s.t.
! 2. Each student is assigned only to one timeslot, instructor, plane
forall (i in I, n in N) sum(j in J, z in Z) x(i,n,j,z) <= 1

! 3. Each plane is assigned to one instructor, student, timeslot
forall (z in Z, n in N) sum(i in I, j in J) x(i,n,j,z) <= 1

!4. and 5. Limit of students per instructor for each timeslot
forall (i in I, n in N, j in J, z in Z) do
        y (i,n,j,z) <= x(i,n,j,z)
        y (i,n,j,z) <= b (i,n)
        y (i,n,j,z) >= b (i,n) - (1-x(i,n,j,z))
end-do

!4a.
forall (n in N, j in J) sum(i in I, z in Z) y(i,n,j,z)<= 1

!Limit of students per instructor for each timeslot in simplified version
        !forall (n in N, j in J) sum(i in I, z in Z) x(i,n,j,z)<= 1

! 6. Only the available timeslots are considered
forall(i in I, n in N, j in J, z in Z) x(i,n,j,z) <= AVAL(i,n)

!7. Max amount of working hours per instructor per month

        forall (j in J) sum (n in N) d (j,n) <= NN/12*8

! 7a. Costs for teachers appear only if a teacher was actually assigned to a
timeslot

forall (n in N, j in J) do
        c(23):= d(j,n) = 1
        c(24):= d (j,n) = 0
        c(25):= sum (i in I, z in Z) x(i,n,j,z) >= 1
        c(26):= sum (i in I, z in Z) x (i,n,j,z) = 0
implies (c (25), c (23))
implies (c (26), c (24))
end-do

!8. Max amount of consecutive flying hours for students during one day
forall(i in I, n in N | n <= NN-S, r in 0..20-1)

sum(j in J, f in n..n+S | f >= 1 + 12*r and f <= 12 + 12*r, z in Z) x(i,f,j,z) <= S

! 9. Determine the total completed hours for each student for each time slot
forall(i in I, a in N) beta(i,a) = sum(j in J, z in Z, n in 1..a) x(i,n,j,z) + PH(i)
```

```
!10. If student finished the last course, don't assign him any more
forall(i in I, n in N) sum(j in J, z in Z) x(i,n,j,z) <= 142 - beta(i,n)

! 11. For the last course, only the 3rd plane can be used
forall(i in I, n in N) do
        c(15):= beta (i,n) >= 135
        c(16):= beta (i,n) <= 141
        c(17):= sum (j in J) x(i,n,j,"C142S") + sum (j in J) x(i,n,j,"C142R")= 0
implies(c(15) and c(16), c(17))

!12. Pick b - how many students per instructor per timeslot?
forall (n in N, i in I) do
        ! For these courses instructor can be assigned to only 1 student
        c(1):= beta (i,n) >= 0
        c(2):= beta (i,n) <= 20

        c(3):= beta (i,n) >= 31
        c(4):= beta (i,n) <= 35

        c(5):= beta (i,n) >= 76
        c(6):= beta (i,n) <= 141

        c(7):= b(i,n) <=1
        c(8):= b(i,n) >=1
implies ((c(1) and c(2)) or (c(3) and c(4)) or (c(5) and c(6)), c(7) and c(8))

! For these courses one instructor can be assigned to max. 3 students
        c(9):= beta (i,n) >= 21
        c(10):= beta (i,n) <= 30

        c(11):= beta (i,n) >= 36
        c(12):= beta (i,n) <= 75

        c(13):= b(i,n) <=0.33
        c(14):= b(i,n) >=0.33
implies ((c(9) and c(10)) or (c(11) and c(12)), (c(13)) and c(14))
end-do

! 13. One of teachers should not teach advance lectures
forall(i in I, n in N) do
        c(18):= beta(i,n) >= 81
        c(19):= sum(z in Z) x(i,n,"Rall",z) = 0
implies(c(18), c(19))
end-do

! 14. For the night course only 18-20(ends 21) hours are allowed
forall(i in I, n in N | (n-1) mod 12 <= 8) do
        c(20):= beta(i,n) >= 76
        c(21):= beta(i,n) <= 80
        c(22):= sum(j in J, z in Z) x(i,n,j,z) = 0
implies(c(20) and c(21), c(22))
```

```
end-do

! Cancel the auxiliary constraints
forall (m in 1..26) c(m) := 0

! 15. Decision var x is binary
forall(i in I, n in N, j in J, z in Z)
        x(i,n,j,z) is_binary

maximize(Y)


! Calculate a course number for each student, each timeslot
forall(i in I, n in N) do
        if(getsol(beta(i,n)) >= 1 and getsol(beta(i,n)) <= 10) then
                q_course(i,n) := 1
        end-if
        if(getsol(beta(i,n)) >= 11 and getsol(beta(i,n)) <= 20) then
                q_course(i,n) := 2
        end-if
        if(getsol(beta(i,n)) >= 21 and getsol(beta(i,n)) <= 30) then
                q_course(i,n) := 3
        end-if
        if(getsol(beta(i,n)) >= 31 and getsol(beta(i,n)) <= 35) then
                q_course(i,n) := 4
        end-if
        if(getsol(beta(i,n)) >= 36 and getsol(beta(i,n)) <= 75) then
                q_course(i,n) := 5
        end-if
        if(getsol(beta(i,n)) >= 76 and getsol(beta(i,n)) <= 80) then
                q_course(i,n) := 6
        end-if
        if(getsol(beta(i,n)) >= 81 and getsol(beta(i,n)) <= 83) then
                q_course(i,n) := 7
        end-if
        if(getsol(beta(i,n)) >= 84 and getsol(beta(i,n)) <= 108) then
                q_course(i,n) := 8
        end-if
        if(getsol(beta(i,n)) >= 109 and getsol(beta(i,n)) <= 109) then
                q_course(i,n) := 9
        end-if
        if(getsol(beta(i,n)) >= 110 and getsol(beta(i,n)) <= 134) then
                q_course(i,n) := 10
        end-if
        if(getsol(beta(i,n)) >= 135 and getsol(beta(i,n)) <= 141) then
                q_course(i,n) := 11
        end-if
end-do

! Show the resulting timetable on the screen (output)
! Head of the table
```

```
writeln("\n\t\t\tTimetable\n-------------------------------------------------------------------
----")
writeln(strfmt("Timeslot", 0), strfmt("Hour", 6), strfmt("Student", 9),
strfmt("Instructor", 12), strfmt("Plane", 8), strfmt("Hours_comp.", 13),
strfmt("Course", 8))
writeln("-------------------------------------------------------------------")

! Content of the table
writeln("Day 1")
forall(n in N) do
        if (n>=13 and (n-1) mod 12 <= 0 and (n-1) mod 12 >= 0) then
                writeln("Day ", (n-1)/12 + 1)
        end-if
        forall(i in I, j in J, z in Z)
                if (getsol(x(i,n,j,z)) >= 1) then
                        writeln(strfmt(n, 8), strfmt((n-1) mod 12 + 1 + 8, 6),
strfmt(i, 9), strfmt(j, 12), strfmt(z, 8), strfmt(getsol(beta(i,n)), 13),
strfmt(q_course(i,n), 8))
                end-if
end-do

end-model
```

APPENDIX B: AN EXAMPLE OF A 5-DAY TIMETABLE

Timeslot	Hour	Student	Instructor	Plane	Hours_comp.	Course
Day 1						
1	9	3	Hartmann	C142R	42	5
2	10	7	Rall	C142R	1	1
2	10	8	Barkhorn	C142S	1	1
3	11	7	Hartmann	C142S	2	1
3	11	8	Rall	C142R	2	1
4	12	7	Rall	C142R	3	1
4	12	8	Barkhorn	C142S	3	1
5	13	7	Hartmann	C142R	4	1
5	13	8	Rall	C142S	4	1
6	14	3	Rall	C142R	43	5
7	15	3	Rall	C142R	44	5
8	16	3	Rall	DA42	45	5
8	16	4	Rall	C142S	42	5
8	16	5	Barkhorn	C142R	1	1
9	17	3	Barkhorn	DA42	46	5
9	17	4	Barkhorn	C142S	43	5
9	17	6	Rall	C142R	1	1
10	18	3	Barkhorn	DA42	47	5
10	18	4	Barkhorn	C142R	44	5
10	18	10	Rall	C142S	76	6
11	19	4	Barkhorn	C142R	45	5
11	19	5	Hartmann	C142S	2	1
11	19	10	Rall	DA42	77	6
12	20	3	Rall	C142R	48	5
12	20	10	Barkhorn	C142S	78	6
Day 2						
13	9	3	Hartmann	C142R	49	5
14	10	7	Rall	C142R	5	1
14	10	8	Hartmann	C142S	5	1
15	11	7	Barkhorn	C142S	6	1
15	11	8	Hartmann	C142R	6	1
16	12	7	Rall	C142R	7	1
16	12	8	Hartmann	C142S	7	1
17	13	7	Rall	C142S	8	1
17	13	8	Barkhorn	C142R	8	1
18	14	3	Barkhorn	C142R	50	5
19	15	3	Rall	C142R	51	5
20	16	3	Barkhorn	DA42	52	5
20	16	4	Barkhorn	C142R	46	5
20	16	5	Hartmann	C142S	3	1
21	17	4	Kozhedub	C142S	47	5
21	17	5	Hartmann	C142R	4	1
21	17	6	Barkhorn	DA42	2	1
22	18	3	Kozhedub	C142R	53	5
22	18	4	Kozhedub	C142S	48	5
22	18	6	Rall	DA42	3	1
23	19	3	Hartmann	DA42	54	5
23	19	4	Hartmann	C142R	49	5
23	19	10	Rall	C142S	79	6
24	20	3	Rall	C142R	55	5
24	20	10	Barkhorn	C142S	80	6

Day 3						
25	9	3	Rall	C142R	56	5
25	9	10	Barkhorn	C142S	81	7
26	10	7	Rall	DA42	9	1
26	10	8	Barkhorn	C142S	9	1
26	10	10	Hartmann	C142R	82	7
27	11	7	Hartmann	C142S	10	1
27	11	8	Rall	C142R	10	1
28	12	7	Rall	DA42	11	2
28	12	8	Hartmann	C142S	11	2
28	12	10	Barkhorn	C142R	83	7
29	13	7	Rall	C142S	12	2
29	13	8	Barkhorn	DA42	12	2
29	13	10	Hartmann	C142R	84	8
30	14	3	Rall	C142R	57	5
30	14	10	Hartmann	C142S	85	8
31	15	3	Barkhorn	C142R	58	5
31	15	10	Hartmann	C142S	86	8
32	16	3	Kozhedub	DA42	59	5
32	16	4	Kozhedub	C142S	50	5
32	16	6	Rall	C142R	4	1
33	17	3	Rall	C142R	60	5
33	17	4	Rall	DA42	51	5
33	17	6	Barkhorn	C142S	5	1
34	18	3	Rall	C142S	61	5
34	18	4	Rall	C142R	52	5
34	18	6	Hartmann	DA42	6	1
35	19	4	Kozhedub	C142S	53	5
35	19	5	Hartmann	DA42	5	1
35	19	10	Barkhorn	C142R	87	8
36	20	3	Rall	C142R	62	5
Day 4						
37	9	3	Rall	C142R	63	5
37	9	10	Hartmann	C142S	89	8
38	10	7	Hartmann	C142R	13	2
38	10	8	Rall	C142S	13	2
38	10	10	Barkhorn	DA42	90	8
39	11	7	Hartmann	C142R	14	2
39	11	8	Barkhorn	C142S	14	2
40	12	7	Hartmann	DA42	15	2
40	12	8	Rall	C142R	15	2
40	12	10	Barkhorn	C142S	91	8
41	13	7	Rall	C142S	16	2
41	13	8	Hartmann	C142R	16	2
41	13	10	Barkhorn	DA42	92	8
42	14	3	Hartmann	C142S	64	5
42	14	10	Barkhorn	C142R	93	8
43	15	3	Rall	C142R	65	5
43	15	10	Kozhedub	C142S	94	8
44	16	3	Hartmann	C142S	66	5
44	16	4	Hartmann	DA42	54	5
44	16	6	Kozhedub	C142R	7	1
45	17	4	Rall	C142R	55	5
45	17	5	Kozhedub	DA42	6	1
45	17	6	Barkhorn	C142S	8	1
46	18	3	Hartmann	DA42	67	5
46	18	4	Hartmann	C142S	56	5
46	18	5	Barkhorn	C142R	7	1
47	19	3	Hartmann	C142R	68	5
47	19	4	Hartmann	DA42	57	5

	48	20	3	Hartmann	C142S	69	5
	48	20	10	Barkhorn	C142R	95	8
Day 5							
	49	9	3	Hartmann	C142R	70	5
	49	9	10	Barkhorn	C142S	96	8
	50	10	7	Hartmann	C142S	17	2
	50	10	8	Rall	DA42	17	2
	50	10	10	Barkhorn	C142R	97	8
	51	11	7	Rall	C142R	18	2
	51	11	8	Hartmann	C142S	18	2
	52	12	7	Barkhorn	C142S	19	2
	52	12	8	Rall	DA42	19	2
	52	12	10	Hartmann	C142R	98	8
	53	13	7	Hartmann	C142S	20	2
	53	13	8	Rall	DA42	20	2
	53	13	10	Barkhorn	C142R	99	8
	54	14	3	Hartmann	C142S	71	5
	54	14	10	Barkhorn	C142R	100	8
	55	15	3	Hartmann	C142S	72	5
	55	15	10	Barkhorn	C142R	101	8
	56	16	4	Rall	DA42	58	5
	56	16	5	Barkhorn	C142S	9	1
	56	16	10	Hartmann	C142R	102	8
	57	17	3	Barkhorn	C142R	73	5
	57	17	4	Barkhorn	DA42	59	5
	57	17	6	Kozhedub	C142S	9	1
	58	18	3	Hartmann	C142R	74	5
	58	18	4	Hartmann	DA42	60	5
	58	18	5	Rall	C142S	10	1
	59	19	3	Barkhorn	DA42	75	5
	59	19	4	Barkhorn	C142S	61	5
	59	19	5	Hartmann	C142R	11	2
	60	20	3	Hartmann	C142S	76	6
	60	20	10	Barkhorn	C142R	103	8

CURRICULUM VITAE

Maria Golova maria.golova@gmail.com

Education

2010 – present	**MSc in Business Administration**, majors: Transportation Logistics and International Marketing, *University of Vienna* Master thesis: "Air fleet assignment model: an example of a flight school timetable formulation"
09.2011 – 01.2012	**Erasmus-Exchange** with specialization in Marketing and Operations Research, *ISCTE Business School Lisbon*
2009 – 2011	**MA in Intercultural Management and Leadership**, *Lauder Business School* (Vienna) Master thesis: "Stepping out of niche markets to mass-markets on an example of the Kosher and Halal food and beverage industry in Austria"
2006 – 2009	**BSc in Business Administration**, majors: Marketing and OR, *Free University of Berlin*
2000	Exchange scholar in California (USA)
1994 – 2006	**Gymnasium**, *Tallinn Jewish School* (Estonia)

Working experience

07.2013 – present	Group Fitness Trainer, Holmes Place GmbH (Vienna)
09.2010 – present	Private tutoring for children (math, science, English) (Vienna)
05.2009 – 07.2009	Internship in Logistics and Warehouse Management, *Kaubaekspress AG* (Estonia)
11.2008 – 01.2009	Student Assistant in Business Information and Controlling, *Creditreform Wolfram KG* (Berlin)

Additional skills

Languages:	English, German, Estonian – fluent; Russian – native
IT skills:	MS Office (Word, Excel, PowerPoint, Visio, Project), SPSS, Xpress (Mosel), Photoshop
Other:	Semi-professional dancer and dance teacher