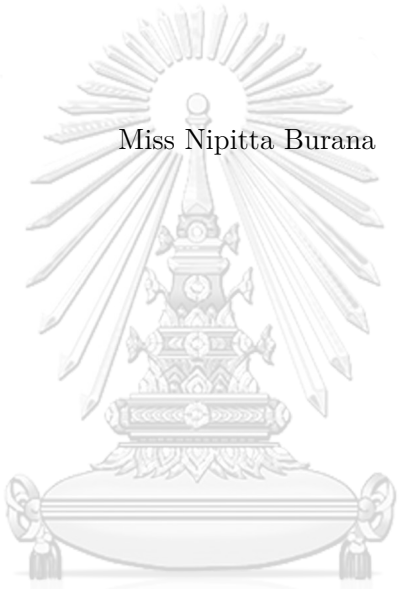
โมเดลสองขั้นสำหรับการปรับสมดุลภาระงานและความชอบในการสอนของผู้สอนในการจัดตารางสอนมหาวิทยาลัย

นางสาวนิพิษฐา บูรณะ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

A TWO-STAGE MODEL FOR BALANCING INSTRUCTOR WORKLOAD AND TEACHING PREFERENCE IN UNIVERSITY COURSE TIMETABLING

Miss Nipitta Burana



A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2020

Thesis Title	A TWO-STAGE MODEL FOR BALANCING INSTRUCTOR WORKLOAD AND TEACHING PREFERENCE IN UNIVERSITY COURSE TIMETABLING
By	Miss Nipitta Burana
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Associate Professor Phantipa Thipwiwatpotjana, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

..... Dean of the Faculty of Science
(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Assistant Professor Krung Sinapiromsaran, Ph.D.)

..... Thesis Advisor
(Associate Professor Phantipa Thipwiwatpotjana, Ph.D.)

..... Examiner
(Assistant Professor Boonyarit Intiyot, Ph.D.)

..... External Examiner
(Associate Professor Chulin Likasiri, Ph.D.)

นิพิษฐา บูรณะ : โมเดลสองขั้นสำหรับการปรับสมดุลภาระงานและความชอบในการสอนของผู้สอนในการจัดตารางสอนมหาวิทยาลัย. (A TWO-STAGE MODEL FOR BALANCING INSTRUCTOR WORKLOAD AND TEACHING PREFERENCE IN UNIVERSITY COURSE TIMETABLING) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : รศ.ดร.พันทิพา ทิพย์วิวัฒน์พจนา 121 หน้า.

วิทยานิพนธ์นี้ชี้ให้เห็นถึงความสมดุลของภาระงานของอาจารย์และความชอบในการสอนโดยใช้ข้อมูลในภาคการศึกษาแรกของปี 2562 จากภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เป็นกรณีศึกษา เนื่องจากอาจารย์หลายคนในภาควิชาของเรามีภาระงานมากเกินไปและปริมาณภาระงานที่มากเกินไปส่งผลกระทบโดยตรงต่อคุณภาพการสอนและการวิจัยของอาจารย์ การสร้างความสมดุลของภาระงานการสอนจึงเป็นจุดประสงค์ของงานนี้ โดยแนวทางในการปรับสมดุลปริมาณภาระงานคือการแบ่งวิชาพื้นฐานออกเป็นสองส่วน: ก่อนกลางภาคและหลังกลางภาค แล้วมอบหมายแต่ละวิชาให้ผู้สอนสองคน นอกจากนี้ความชอบหรือความสามารถในการสอนวิชาก็มีความสำคัญในการทำให้ผู้สอนรู้สึกอุ่นใจ และยังช่วยให้นักเรียนได้รับความรู้อย่างเต็มศักยภาพอีกด้วย ผลลัพธ์ของงานวิจัยนี้แสดงให้เห็นว่าโมเดลของเราสามารถลดจำนวนอาจารย์ผู้ที่มีความต่างสูงของภาระที่ร้องขอกับภาระงานที่ได้รับและจำนวนวิชาที่อาจารย์ไม่ต้องการสอนได้โดยการเปรียบเทียบผลลัพธ์ที่ได้จากงานนี้กับตารางเรียนของภาควิชาและแบบจำลองที่ไม่มีการแบ่งวิชา

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา	ลายมือชื่อ อ.ที่ปรึกษาหลัก	.....
ปีการศึกษา	2563		

6171992023 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : UNIVERSITY TIMETABLING PROBLEM / INTEGER PROGRAMMING / OPTIMIZATION

NIPITTA BURANA : A TWO-STAGE MODEL FOR BALANCING INSTRUCTOR WORKLOAD AND TEACHING PREFERENCE IN UNIVERSITY COURSE TIMETABLING.
ADVISOR : ASSOC. PROF. Phantipa Thipwiwatpotjana, Ph.D., 121 pp.

This thesis focuses on balancing instructor workload and maximizing preferences by using the data in the first semester of 2019 from Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University as a case study. Since there are many instructors with over-workload in the department which directly affect their research qualities, balancing teaching workload is the main objective of this study. The proposed approach to balance workload is to split some basic courses into two parts: before midterm and after midterm and then assign each course to two instructors. Moreover, the preferences or the requests of teaching a course are important to maintain the instructor comfortable, and the preferences or the requests of teaching a course also help students to gain knowledge to their full potentials. The results show that our model is able to reduce both the number of instructors who have the high difference of requested workload and assigned workload, and the number of non-preferable courses for each instructor by comparing our results with the department timetable and the model without splitting courses.

Department	: Mathematics and Computer Science	Student's Signature	.....
		Advisor's Signature	.....
Field of Study	: Applied Mathematics and Computational Science		
Academic Year	: 2020		

ACKNOWLEDGEMENTS

This thesis could not have been achieved without the help from the following important people and project.

Firstly, I would like to thank my advisor, Associate Professor Phantipa Thipwiwatpotjana, Ph.D, for giving me the motivation for this thesis, basic background and suggestions.

Moreover, I would like to appreciate the great help of Assistant Professor Boonyarit Intiyot, Ph.D, Assistant Professor Krung Sinapiromsaran, Ph.D, Associate Professor Chulin Likasiri, Ph.D, and Associate Professor William G. Ferrell, Ph.D. They provide the valuable comments and advises for me to complete my thesis successfully.

Special thanks are given to my teachers in the faculty of Science and the Department of Mathematics and Computer Science for their supports during my education in Chulalongkorn University. Finally, I would like to thank my family and friends very much for supporting me.

This thesis is funded by Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, and the Development and Promotion of Science and Technology Talents Project (DPST).

CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Is anyone in the world living without schedules? As soon as we open our eyes, we think about what we are going to do for the day. Business people need to know when and where they might have meetings. Students need to know when to wake up in order to be in school on time. After that, students need to know what subjects they will study and when and where to do it. Moreover, scheduling is important in many fields: transportation, machine operation, nursing, education, and many others.

As students at the university, we saw difficulty of course scheduling. At the beginning of a semester, each department must have courses scheduled for the term so that students can decide which courses to enroll in without courses overlapping. What can the head of the department do to prevent students from having difficulty in trying to enroll desired courses without such overlap? If the head of the department can solve these scheduling problems, students will be able to sign up for the courses they are interested in instead of having to make difficult choices between courses that are offered at the same time.

Moreover, there are some other conflict among courses scheduling using a method. Some of the courses are scheduled in the same classroom at the same time slot. Therefore, this thesis intends to create effective course scheduling in order to decrease these mistakes and still satisfy all departmental restrictions and needs. Only courses from the first semester of 2019 were provided by Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University to be used in this study. Every academic year, the department conducts a survey of teaching preferences for each course and also the amount of requested workload

for each semester of each instructor. The responsible staff at the department usually organizes the course schedule by hand, using the previous year's timetable. That timetable might not have been efficient because of curriculum changes, new courses, retired faculties from time to time. Moreover, many instructors had work overload which directly affected their teaching and research qualities. Therefore, this thesis aims to obtain course scheduling practices that satisfy both teaching preferences and assigned workload which as close as possible to the requested by instructors. The constraints are under the departmental restrictions. In addition, if some courses can be split into two portions: before midterm and after midterm, and instructors can share each part to teach, it might help balance the teaching workload.

This idea of splitting some basic courses into two parts is the main focus for our model. The model consists of two stages: the first stage model and the second stage model. The first stage model is to regularly solve the timetable by considering all department restrictions. Since there are a large number of constraints, constraint programming approach is quite suitable tool for this stage. The classroom and time slot should be the same in both parts of split courses. So, the second stage model is to match only instructors with their preferred courses. This model is called the two-stage model. After solving the two-stage model by using CPLEX optimization studio version 12.8.0, we obtain the full timetable which satisfies all department restrictions. The first stage model runs on a computer with an Intel Xeon Platinum CPU and 127 GB of RAM. The second stage model runs on a laptop with an Intel Core i7 CPU and 8 GB of RAM. The timetable matches instructors with their assigned courses in suitable classrooms at appropriate time slots. Moreover, the number of instructors in each difference range between the requested and the assigned workload of our two-stage model are compared with the timetable which obtained by department and the first stage model alone. The

number of assigned courses in each preference rank of our two-stage model is also compared with the other two models as well.

This thesis is presented in five chapters, starting with an introduction in Chapter I. Literature reviews and background knowledge about constraint programming (CP) are described in Chapter II. Chapter III explains required information for the model. The model for solving course scheduling presents in Chapter IV. The last chapter (Chapter V) shows the result that how our two-stage model helps balancing instructor workload and preferences, and gives the conclusions of our work.

CHAPTER II

BACKGROUND KNOWLEDGE AND LITERATURE REVIEWS

This chapter presents a general constraint programming (CP) concept and reviews the relevant research papers with a list of constraints on each paper.

2.1 Constraint programming (CP)

Constraint programming (CP) is a program for dealing with the complexity of real-world problems especially finding solutions of scheduling problems: for example, people scheduling, machines scheduling, vehicles scheduling. A CP model consists of five parts: vocabularies definition, decision variables, search setup, objective function, and constraints. It may also have post-processing expressions at the end of a model. Defining vocabularies is the first part for letting the program know what does each word represents. Decision variables are the unknown information in a problem and constraints are the restrictions on combinations of these decision variables. A CP model also contains an objective that can be minimized or maximized. Search setup part is to determine the limit of calculation length and to select the search type. There are three search types in CPLEX optimizer: restart search, depth-first search, and multi-point search.

Restart search is a search procedure that restarts from time to time and leads to an optimal solution. In addition, this search is the default search in CPLEX.

Depth-first search is a tree search algorithm that works on one branch of the subtree until it found a solution or has no solution in that subtree. The optimizer will not move to work on another branch until the current one has been fully explored.

Multi-point search is a search that runs until the optimizer found the best solution that it cannot improve. So, it should set up a limit time when using this search.

However, the search type can be changed depending on models. The restart search is applied in our two-stage model because this search obtains the best results compared to the other two searches with the maximum calculation time setting.

2.2 Literature reviews

Scheduling is important in education fields such as school scheduling, examination scheduling, and university scheduling. There is a slight difference between the school schedule and the university schedule, since most subjects in high school or below have already been assigned to the students. Students cannot choose what they are interested in. There are some school scheduling articles trying to solve the problem using an approach suitable for their model. For example, C. Valouxis and E. Housos [15] present solving high school timetabling problem by using constraint programming (CP). O. A. Odeniyi *et al.* [13] consider both mathematical programming and enhanced simulated annealing algorithm for solving the school timetabling problem. For an examination schedule, B. Genc and B. O'Sullivan [4] consider a two-phase constraint programming model for solving examination timetabling at University College Cork. The first phase considers the timing of examinations while the second phase considers their exam room allocation.

Many researchers are interested in the university course schedule since there

are many interesting points such as various constraints and objectives, and methods for finding the suitable solutions. Some research papers only deal with assigning courses to the suitable instructors such as [3], [14], and [16]. B. Domench and A. Lusa [3] present a mixed integer model for the instructor assignment problem by focusing on instructor's preferences. P. Thipwiwatpotjana [14] concerns about an uncertainty information in term of interval requested workload. After that, [16] considers instructor fuzzy satisfaction in a teaching course assignment problem. In addition, many research papers consider a part of assigning student groups to courses at reasonable time slot in suitable classroom, for example, [1] and [9]. These two papers use different methods for solving the problem, constraint programming is used in [9] but a hybrid meta-heuristic approach is used in the other. Not only the researches mentioned above, but there are also several studies have tried to obtain the full timetable which includes assigning instructors to course sections for student groups at appropriate time slot in suitable classrooms. A. Gunawan *et al.* [7] present the full timetable by using a hybridized Lagrangian relaxation and simulated annealing method for solving the problem. B. Naderi [12] compares the solution which get from three different methods: imperialist competitive algorithm, simulated annealing, and variable neighborhood search. A. A. Gozali *et al.* [6] use localized island model genetic algorithm with dual dynamic migration policy. S. I. Hossain *et al.* [8] use particle swarm optimization with selective search. Moreover, the timetable problem is a complex problem, one of the approach for solving is constraint programming. This approach is widely used in timetable problems as presented in [2], [9], [15], and [17].

Constraint programming (CP) and linear programming (LP) are applied in this thesis. CP is used to find the full timetable and after that we use the results obtained from the first stage model and LP to balance instructor workload in the second stage model. Our two-stage model is derived by applying Department of

Mathematics and Computer Science, Faculty of Science, Chulalongkorn University restrictions to the model from research papers. We summarize all objective functions and constraints which mentioned in literature review in Table 2.1 and Table 2.2, respectively. The checked marks in the second column of both tables are objectives and constraints used in our two-stage model.

Objectives	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
1. Maximizing the teaching preferences of instructors.	✓		✓		✓			✓	✓	✓	✓	✓
2. Maximizing the total preference of instructor-day and course-day.							✓					
3. Balancing instructors teaching load.	✓								✓			
4. Balancing instructor workload.	✓									✓		
5. Minimizing the number of course credits assigned to the instructors.												✓
6. Minimizing the working days of instructors.												✓
7. Minimizing the working days of students.												✓
8. Minimizing the instructors idle time.												✓
9. Minimizing the students idle time.												✓
10. Minimizing the overlapping conflicts between two courses.												✓
11. Satisfying all the hard constraints.		✓		✓		✓	✓					
12. Minimizing the violation of the soft constraints.		✓		✓			✓					

Table 2.1: Various objectives of course scheduling models from research articles.

Constraints	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
Instructor												
1. An instructor can teach at most one section at any time slot.	✓	✓		✓	✓		✓	✓				✓
2. Instructors can teach the assigned courses, based on their preferences.	✓		✓						✓			
3. A course (section) should be taught by only one instructor.	✓		✓								✓	
4. Every instructor should meet his/her own requested amount of workload.										✓		
5. Each instructor can only teach at most one section for each course.	✓	✓								✓		
6. Each instructor allows to teach the limited number of courses.	✓							✓				
7. There are the maximum number of teaching hours for each instructor in each working day.	✓			✓				✓				

continued ...

...continued

	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
8. There are the maximum number of courses for each instructor in each working day.									✓			✓
9. Instructors should not teach too many consecutive time slots.					✓							
10. Some instructors should be scheduled in their preferred time slot.				✓								
11. Instructors should have available time more than their requested between two teaching courses.				✓								
12. If an instructor of a specific course is not available at a given time slot, then no lecture of the course can be scheduled at that time slot.		✓										✓
13. The total teaching activity points assigned to an instructor for a year must be between the preset min and max values.			✓									
14. Each course section is taught by one instructor.	✓		✓									

continued ...

...continued

	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
15. The absolute value of a teaching activity point is underload or overload for each instructor.	✓		✓									
16. The number of courses assigned to the invited instructor should be more than one in a day.								✓				
Classroom												
17. Each classroom is used for at most one course section at any time slot.	✓	✓				✓	✓					
18. The classroom capacity must be able to serve the number of students in the assigned course.	✓	✓		✓	✓		✓		✓			✓
19. The classroom must be able to support laboratory and/or lecture courses.	✓				✓	✓		✓	✓			
20. All lectures of a course should be delivered in the same classroom.		✓										

continued ...

...continued

	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
Student group												
21. Students in the same group study at most one course (section) at any time slot.	✓			✓	✓	✓	✓					
22. All courses of a given student group must be scheduled in different time slots.	✓			✓								✓
23. There are a minimum number of lectures assigned to the student groups in a working day.				✓								
24. There are a maximum number of lectures assigned to the student groups in a working day.												✓
25. A student should not attend more than two consecutive classes.							✓					
26. There are a minimum and maximum number of days assigned courses of a student group.								✓				

continued ...

...continued

	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
Course												
27. Common courses are not offered at the same time slot.	✓							✓				
28. A course starts and ends in either the first half or the second half of the day.	✓											
29. Courses cannot be assigned to break durations e.g. lunch time.	✓				✓							
30. If a course has more than one meeting per week, the course meeting cannot be assigned to the time slots at the same day or on the consecutive days.	✓											✓
31. Some courses have break duration between specified courses.	✓											
32. No classes on Wednesday afternoon.	✓											
33. Each type of course should be assigned to a predefined period of time slots. For example, some courses must be assigned to teach in the morning or the afternoon.	✓					✓						✓

continued ...

...continued

	Our model	[1]	[3]	[6]	[8]	[9]	[10]	[11]	[12]	[14]	[16]	[18]
34. Each course should start at the beginning of an hour.	✓											
35. The time interval between two classes for group teaching should less than its minimum time constraint.				✓								
36. Some course lectures should be adjacent to each other.		✓										

Table 2.2: The constraints of course scheduling models from research articles.

CHAPTER III

REQUIRED INFORMATION FOR THE MODEL

This chapter lists the required information and how it appears in Excel files. The data used in this work is from the first semester of 2019 provided by Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University. Our department contains 60 instructors, 86 courses. These 86 courses turn into 122 course sections. There are 53 classrooms with various sizes. Classes start from 8:00 AM – 12:00 PM and 1:00 – 5:00 PM in every working day by dividing into 16 half-hour time slots.

3.1 Teaching preferences

In every academic year, the head of the department asks each instructor to provide his/her preference ranking of each course as shown in Figure 3.1 or for more details in Appendix A. A. Gorka and P. Thipwiwatpotjana [5] study about the important preference ranking value. In our work, we will use the same value of preference ranking as [5] and teaching preference description is provided in Table 3.1. After that, we change the preference ranking data in Figure 3.1 to the values in Table 3.1 then weight them with 100 to present them as integer in Table 3.2. The full changed and weighted teaching preference values of each instructor towards all courses are presented in Appendix B.

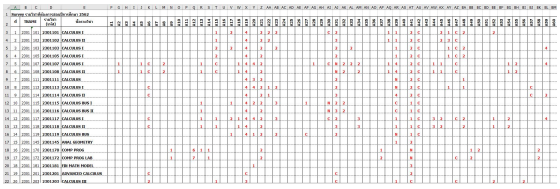


Figure 3.1: A survey of preference ranking of each course.

Preference descriptions	Rank	Values of preferences
A number one preferable course to teach	1	1.00
A preferable course to teach	2	0.95
An ok course to teach	3	0.90
A non preferable course but ok to teach	4	0.85
A non preferable course	C	0.80
A non preferable course and does not want to teach	N, N/A	N/A

Table 3.1: Teaching preference description.

Instructors	Courses	Values of weighted preferences
A1	2301386	100
A2	2301107	100
A2	2301481	100
A2	2301520	100
A2	2301591	90
⋮	⋮	⋮
A60	2301312	100

Table 3.2: An example of the full changed and weighted teaching preference values.

3.2 Workloads

The head of the department asks each instructor to provide his/her requested workload at the beginning of every academic year including seminar and advisor workload as in Table 3.3 or the attached link in Appendix C for the Excel file. The total workload of each instructor consists of three parts: the teaching workload,

the seminar workload, and the advisor workload. The teaching workload of a course depends on the number of students enrolled and the credit of the course. If the number of students is up to 50, the teaching is 3 times credits of the course. If the number of students is more than 50 and up to 100, the teaching is 3.5 times credits of the course. If the number of students is more than 100 and up to 150, the teaching is 4 times credits of the course. And if the number of students is over 150, the teaching is 4.5 times credits of the course. The estimated number of students enrolled, the credits, and the workload for each course are shown in Table 3.4, for an example or the attached link in Appendix D for more details.

Instructors	Maximum teaching hours per day	Workloads		
		Requested	Seminar	Advisor
A1	8	13.50	0.00	4.50
A2	8	24.50	2.00	10.50
A3	8	21.00	0.00	4.50
⋮	⋮	⋮	⋮	⋮
A60	8	21.00	3.27	3.00

Table 3.3: An example of the number of maximum teaching hours per day, requested, seminar, and advisor workloads of instructors.

Courses	#Students	Workloads	Credits
2301101	103	16.00	4
2301103	100	10.50	3
2301107	89	10.50	3
⋮	⋮	⋮	⋮
2301736	15	9.00	3

Table 3.4: An example of the number of students, workload, and credits for each course.

3.3 Student groups

We divide students into groups by their faculty, study field, and study level (year) so that the timetables in the student groups do not overlap. There are 22

student groups as the following. The first 11 groups (including Math-1, Math-2, Math-3, Math-4, Comp-1, Comp-2, Comp-3, Comp-4, Science-1, Science-2, and Science-3) are students in the faculty of Science and 11 other groups are pharmaceutical science, education, engineering, agriculture resources, and commerce and accountancy students (Pharm Sci, Pharm Care, Education-1, Education-2, Education-3, Engineering-1, Engineering-2, Ocare, Management, Account, and Statistics).

3.4 Classrooms

There are two types of classrooms: classrooms in faculty of Science and in other faculties. Students should study in their faculty classroom. The example of classrooms and their capacities are shown in Table 3.5. The full lists of classrooms and their capacities are attached in Appendix E. There are five different classrooms: lab rooms, small rooms, medium rooms, large rooms, and dummy rooms. Since we do not know how many non-science classroom are there and how much the capacity of these rooms is, we will entrust their faculties to set classrooms for their students. In order to continue our work, we set 15 dummy rooms as classroom numbers 39 - 53 with the largest capacity (300 students).

3.5 Time slots

We consider teaching time on five working days. Since some courses are taught in one and a half hour at a time, so we divide time interval into 16 time slots a day from 8:00 AM - 12:00 PM and 1:00 - 5:00 PM or 80 time slots a week starting with 0^{th} time slot by 1 time slot means a half hour. For example, the 0^{th} time slot means 8:00 - 8:30 AM on Monday and the 79^{th} time slot is the last half an hour on Friday. Moreover, the department has a restriction that there are no classes on Wednesday afternoon due to the activities of the department which

No.	Room types	Classrooms	Capacity
1.	Lab	MATH-508/1	42
2.		MATH-509/2	76
3.		MATH-708/5	50
4.	Small	MATH-608/6	20
5.		MATH-608/8	20
⋮		⋮	⋮
10.		MATH-1008B	24
11.	Medium	MATH-809/3	30
12.		MATH-809/4	30
⋮		⋮	⋮
26.		TAB-231	40
27.	Large	MHMK-M01	284
28.		MHMK-M02	284
⋮		⋮	⋮
38.		TAB-222	250
39.	Dummy	d1	300
40.		d2	300
⋮		⋮	⋮
53.		d15	300

Table 3.5: An example of classrooms and their capacities.

are reserved for department meetings. Therefore, the $40^{th} - 47^{th}$ time slots are unavailable teaching time slots.

3.6 Courses

Courses in our department are assigned to meet more than once a week and the teaching period of some courses may not necessarily equal in every meeting. For example, "2301442" course requires 2 and 4 time slots in one week as shown in Table 3.6. Required duration and its repetition of all courses are described in Appendix F. Moreover, some specific courses need to be scheduled in specific time slot. Calculus and basic computer programming courses are always scheduled in the morning and some courses should be taught in the afternoon as shown in Table 3.7. Moreover, some specific courses cannot be taught in consecutive time slots

or need a break between two courses as shown in the same table. For example, NeedBreak courses (2301217, 2301224) means course 2301217 and course 2301224 should be apart for at least one time slot.

Student groups	Courses	Sections	Duration (#Time slots)	Repetition (#Times)
Math-1	2301117	1	2	4
Math-4	2301442	1	2	1
Math-4	2301442	1	4	1
⋮	⋮	⋮	⋮	⋮
Comp M.Sc	2301736	1	6	1

Table 3.6: An example of required duration and its repetition of all courses.

Morning courses	Afternoon courses	NeedBreak courses
2301101	2301233	(2301217, 2301224)
2301103	2301361	(2301234, 2301224)
2301107		(2301307, 2301337)
2301113		
2301117		
2301119		
2301170		

Table 3.7: Morning, afternoon courses and courses that need a break.

CHAPTER IV

THE MODEL

This chapter explains the concept of our model. Since instructors should not teach far more than their requested workload, some basic courses similar to Calculus are split into two parts: before midterm and after midterm, in order to assign each part to an instructor. It means that a basic course can be taught by two instructors. With this idea, we will divide the calculation into two stages. The first stage is to regularly solve the model with satisfying all department restrictions by constraint programming. The second stage is to find just the instructor for the basic courses by teaching time slot and the classroom will be the same as before midterm part. If we do not want to split some basic courses, we can solve just the first stage model with the full course credits of teaching workload. Finally, we will get the full timetable and we call this model that the first stage model alone. All notations, parameters, decision variables, objective function, and constraints of the first stage model and the second stage model are presented in Section 4.1 and Section 4.2, respectively. We would like to mention that the models described in this chapter are merely a presentation for the reader to see the relationship of restrictions in mathematical terms. The computational models used in this work are presented in Appendix J and Appendix K.

4.1 The first stage model

The first stage model is to assign the course section to the right instructor with the suitable classroom and the appropriate time slot for each student group. Explanations of notations, parameters, decision variables, objective function, and constraints are presented as follows.

4.1.1 Notations and parameters

Let I be the set of all instructors,

J_i be the set of courses that an instructor i can teach,

J_{lab} be the set of courses that should be taught in a laboratory room,

J_{lec} be the set of courses that should be taught in a lecture room,

J_{am} be the set of morning courses,

J_{pm} be the set of afternoon courses,

J_{break} be the set of two courses that cannot be taught in consecutive time slots,

J be the set of all courses such that $J = \bigcup_{\forall i \in I} J_i = J_{lab} \cup J_{lec}$,

K_j be the set of sections of course j,

L be the set of all student groups,

R_{lab} be the set of all laboratory classrooms,

R_{lec} be the set of all lecture classrooms,

R_{large} be the set of all large classrooms,

R be the set of all classrooms such that $R = R_{lab} \cup R_{lec}$,

T be the set of time slots such that $T = \{0, 1, \ldots, 79\}$,

$T_{j,k,q}$ be the set of time slots of session q of course j section k where $T_{j,k,c} \cap T_{j,k,d} = \emptyset$ s.t. $c \neq d$,

m_i be the maximum number of teaching hours for instructor i in a working day,

c_r be the capacity of classroom r,

n_j be the number of students enrolled in course j in each section, assuming that all sections have the same number of students,

$n_{j,k}$ be the number of sessions in a week of course j section k,

$n_{j,k,q}$ be the durations of session q of course j section k, where $q \in \{1, 2, \ldots, n_{j,k}\}$,

a_i be the advisor workload of instructor i,

b_i be the seminar workload of instructor i,

r_i be the requested workload of instructor i,

w_j be the workload of course j,

and $p_{i,j}$ be the teaching preference value of instructor i who prefers to teach course j.

4.1.2 Decision variables

Let $x_{i,j,k,l,r,t}$ be a binary variable taking value 1 if instructor i teaches course j section k to student group l in classroom r at time slot t, otherwise it equals 0,

$x_{i,j,k}$ be a binary variable taking value 1 if instructor i teaches course j section k, otherwise it equals 0,

$x_{j,k,t}$ be a binary variable taking value 1 if course j section k teaches at time slot t, otherwise it equals 0,

$x_{i,j}$ be a binary variable taking value 1 if instructor i teaches course j, otherwise it equals 0,

s_i be a positive part of the subtraction between the assigned and requested workload for instructor i,

and u_i be a negative part of the subtraction between the assigned and requested workload for instructor i.

4.1.3 Objective function

There are two objectives that we consider in this thesis. The objective functions are presented in terms of weighted sum method in equation (4.1).

$$\min -\alpha_1 f_1 + \alpha_2 f_2 \tag{4.1}$$

where $f_1 = \sum_{i \in I} \sum_{j \in J_i} p_{i,j} x_{i,j}$ is the total teaching preferences,
$f_2 = \sum_{i \in I} \left| (\sum_{j \in J} w_j x_{i,j} + a_i + b_i) - r_i \right| = \sum_{i \in I} (s_i + u_i)$ is the total difference between instructors requested workload and their assigned workload.

We want to maximize the total teaching preference (f_1) while we want to minimize the total difference of assigned workload and requested workload of instructors (f_2). Since we give the same priority for f_1 and f_2, both α_1 and α_2 are equal. For simple calculation, α_1 and α_2 can be set to 1.

4.1.4 Constraints

In order to satisfy our department restrictions, there are 23 constraints that are considered in this model.

1. An instructor can teach at most one course section at any time slot.

$$\sum_{j \in J} \sum_{k \in K_j} \sum_{l \in L} \sum_{r \in R} x_{i,j,k,l,r,t} \leq 1, \quad \forall i \in I, \forall t \in T \tag{4.2}$$

2. Instructors would not assigned courses that they cannot teach.

$$x_{i,j,k,l,r,t} = 0, \quad \forall i \in I, \forall j \in J \setminus J_i, \forall k \in K_j, \forall l \in L, \forall r \in R, \forall t \in T \tag{4.3}$$

3. Each classroom is used for at most one course section at any time slot.

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L} x_{i,j,k,l,r,t} \leq 1, \quad \forall r \in R, \forall t \in T \tag{4.4}$$

4. The assigned classroom must support laboratory and/or lecture courses.

   (a) Lecture courses should not be taught in a laboratory room.

$$x_{i,j,k,l,r,t} = 0, \quad \forall i \in I, \forall j \in J_{lec}, \forall k \in K_j, \forall l \in L, \forall r \in R_{lab}, \forall t \in T \tag{4.5}$$

   (b) Laboratory courses should not be taught in a lecture room.

$$x_{i,j,k,l,r,t} = 0, \quad \forall i \in I, \forall j \in J_{lab}, \forall k \in K_j, \forall l \in L, \forall r \in R_{lec}, \forall t \in T \tag{4.6}$$

5. The classroom capacity must be able to serve the number of students in the assigned course. Assume that each course has the same number of students registered in all sections.

$$c_r \geq n_j x_{i,j,k,l,r,t}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L, \forall r \in R, \forall t \in T \tag{4.7}$$

   In addition, the classroom should not be too large because it will waste resources. Therefore, course sections with fewer than 40 students cannot be use large classrooms.

$$x_{i,j,k,l,r,t} = 0, \quad \forall i \in I, \forall j \in J \text{ s.t. } n_j \leq 40, \forall k \in K_j, \forall l \in L, \forall t \in T, \text{ and } r \in R_{large} \tag{4.8}$$

6. Students who are in the same group study at most one course section at any

time slot.

$$\sum_{i\in I}\sum_{j\in J}\sum_{k\in K_j}\sum_{r\in R} x_{i,j,k,l,r,t} \leq 1, \quad \forall l \in L, \forall t \in T \tag{4.9}$$

7. Each course section (in the first stage model) is taught by one instructor.

$$\sum_{i\in I} x_{i,j,k} = 1, \quad \forall j \in J, \forall k \in K_j \tag{4.10}$$

8. A course section starts and ends in either the first half or the second half of the day.

$$x_{j,k,t_1} + x_{j,k,t_2} \leq 1, \quad \forall j \in J, \forall k \in K_j, \forall t_1, t_2 \in T \text{ s.t. } \left\lfloor \frac{t_1}{8} \right\rfloor = 0, 2, 4, 6, 8, \left\lfloor \frac{t_2}{8} \right\rfloor = 1, 3, 5, 7, 9, \text{ and } \left\lfloor \frac{t_1}{8} \right\rfloor + 1 = \left\lfloor \frac{t_2}{8} \right\rfloor \tag{4.11}$$

Note that $\left\lfloor \frac{t}{8} \right\rfloor = 0, 2, 4, 6, 8$ means time slot in the first half of the day or in the morning. $\left\lfloor \frac{t}{8} \right\rfloor = 1, 3, 5, 7, 9$ means time slot in the second half of the day or in the afternoon.

	The first half of the day		The second half of the day	
	Time slots (t)	$\left\lfloor \frac{t}{8} \right\rfloor$	Time slots (t)	$\left\lfloor \frac{t}{8} \right\rfloor$
Monday	0,1,...,7	0	8,9,...,15	1
Tuesday	16,17,...,23	2	24,25,...,31	3
Wednesday	32,33,...,39	4	40,41,...,47	5
Thursday	48,49,...,55	6	56,57,...,63	7
Friday	64,65,...,71	8	72,73,...,79	9

Table 4.3: All time slots in a week.

9. Some courses have a break (at least a half-hour) between specified courses. For example, if a course "2301217" is taught at time slot 0,1, and 2, then a course "2301224" can be taught at time slot 4 and 5.

$$x_{j_1,k,t} + x_{j_2,k,t+1} \leq 1, \quad \forall j_1, j_2 \in J_{break} \text{ s.t. } j_1 \neq j_2, \forall k \in K_j, \forall t \in T \tag{4.12}$$

10. There are no classes on every Wednesday afternoon.

$$x_{j,k,t} = 0, \quad \forall j \in J, \forall k \in K_j, \text{ and } t \in T \text{ s.t. } t = 40, 41, \ldots, 47 \tag{4.13}$$

11. Some specific courses must be assigned to teach in preferred duration.

    (a) The morning courses cannot be assigned to teach in the afternoon.

$$x_{j,k,t} = 0, \quad \forall j \in J_{am}, \forall k \in K_j, \text{ and } t \in T \text{ s.t. } \left\lfloor \frac{t}{8} \right\rfloor = 1, 3, 5, 7, 9 \tag{4.14}$$

    (b) The afternoon courses cannot be assigned to teach in the morning.

$$x_{j,k,t} = 0, \quad \forall j \in J_{pm}, \forall k \in K_j, \text{ and } t \in T \text{ s.t. } \left\lfloor \frac{t}{8} \right\rfloor = 0, 2, 4, 6, 8 \tag{4.15}$$

12. If a course section has more than one meeting per week, the course meeting cannot be assigned to the time slots at the same day or on the consecutive days except for courses that are taught four times a week.

$$\text{If } n_{j,k} \leq 3, \sum_{\lfloor \frac{t}{16} \rfloor = i}^{\lfloor \frac{t}{16} \rfloor = i+1} x_{j,k,t} \leq n_{j,k,q}, \quad \forall j \in J, \forall k \in K_j, \forall q \in \{1, 2, \ldots, n_{j,k}\},$$
$$\text{and } t \in T_{j,k,q} \text{ s.t. } i = 0, 1, 2, 3 \tag{4.16}$$

13. If course j section k which needs to be taught $n_{j,k}$ sessions in a week needs more than one time slot per session, each session must be assigned on $n_{j,k,q}$ consecutive time slots in a session, where $n_{j,k,q} \geq 2$ and $q \in \{1, 2, \ldots, n_{j,k}\}$.

$$\text{If } x_{j,k,t-1} = 0 \text{ and } x_{j,k,t} = 1, \quad \sum_{i=0}^{n_{j,k,q}-1} x_{j,k,t+i} = n_{j,k,q}, \quad \forall j \in J, \forall k \in K_j,$$
$$\forall q \in \{1, 2, \ldots, n_{j,k}\}, t + i \in T_{j,k,q} \text{ s.t. } i = 0, 1, \ldots, n_{j,k,q} - 1 \tag{4.17}$$

For example, section 1 of the course "2301442" requires 2 sessions a week: 2 time slots and 4 time slots for each session.

$$x_{2301442,1,t} + x_{2301442,1,t+1} = 2, \quad \text{where } t \in T_{2301442,1,1} \tag{4.18}$$

$$x_{2301442,1,s} + x_{2301442,1,s+1} + x_{2301442,1,s+2} + x_{2301442,1,s+3} = 4, \quad \text{where } s \in T_{2301442,1,2} \tag{4.19}$$

14. There are the maximum number of teaching hours for each instructor in each working day.

$$\sum_{j\in J}\sum_{k\in K_j}\sum_{l\in L}\sum_{r\in R}\sum_{t\in T} x_{i,j,k,l,r,t} \leq 2m_i, \quad \forall i \in I, \forall n \in \{0,1,2,3,4\} \text{ s.t. } n = \left\lfloor \frac{t}{16} \right\rfloor \tag{4.20}$$

15. Each instructor can only teach at most one section for each course.

$$\sum_{k\in K_j} x_{i,j,k} \leq 1, \quad \forall i \in I, \forall j \in J \tag{4.21}$$

16. Each course section cannot start at a half-hour or should start at the beginning of an hour. (If a course section is not taught at the beginning of an hour, the remaining of a half-hour will not be able to teach that course section.)

$$\text{If } x_{j,k,t} = 0, \quad x_{j,k,t+1} = 0 \quad \forall j \in J, \forall k \in K_j, \text{ and } t = 0, 2, \ldots, 78 \tag{4.22}$$

17. Each instructor is allowed to teach at most three course sections.

$$\sum_{j\in J}\sum_{k\in K_j} x_{i,j,k} \leq 3, \quad \forall i \in I \tag{4.23}$$

18. Instructor i must be assigned to course j first before being able to know the course section k.

$$x_{i,j} \geq x_{i,j,k}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j \tag{4.24}$$

19. Instructor i must be assigned to the section k of course j before choosing student group l in classroom r and at time slot t.

$$x_{i,j,k} \geq x_{i,j,k,l,r,t}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L, \forall r \in R, \forall t \in T \tag{4.25}$$

20. Course j section k is assigned to student group l in classroom r at time slot t which taught by instructor i must be assigned at time slot t.

$$x_{j,k,t} \geq x_{i,j,k,l,r,t}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L, \forall r \in R, \forall t \in T \tag{4.26}$$

Note $x_{i,j,k}$ and $x_{j,k,t}$ are not related to each other because $x_{i,j,k} = 1$ means instructor i teaches course j section k but $x_{j,k,t} = 1$ means course j section k is taught at time slot t.

21. These constraints will force at least one of s_i and u_i to be zero.

$$s_i \geq (\sum_{j \in J} w_j x_{i,j} + a_i + b_i) - r_i, \quad \forall i \in I \tag{4.27}$$

$$u_i \geq r_i - (\sum_{j \in J} w_j x_{i,j} + a_i + b_i), \quad \forall i \in I \tag{4.28}$$

22. All variables are binary variables.

$$x_{i,j,k,l,r,t},\ x_{i,j,k},\ x_{j,k,t},\ x_{i,j} \in \{0,1\},$$
$$\forall i \in I, \forall j \in J, \forall k \in K_j, \forall l \in L, \forall r \in R, \forall t \in T \tag{4.29}$$

23. The positive part (s_i) and the negative part (u_i) are the subtraction between the assigned and requested workload for each instructor i must be non-negative.

$$s_i,\ u_i \geq 0, \quad \forall i \in I \tag{4.30}$$

Since some courses need to be taught by the same instructor in both before and after midterm part while some (basic) courses can be taught by the different instructors for each part, so these basic courses will be computed to find the other instructor in the second stage model. After solving the first stage model, we have got the non-basic course timetable for the whole semester and the basic courses timetable before midterm part. We will continue to solve the basic course timetable after midterm part in the second stage which will presented in Section 4.2.

4.2 The second stage model

In this stage, we try to find only instructors for teaching basic courses in after midterm part. Note the basic courses should be taught at the same classroom and time slot in both before midterm and after midterm in order to avoid confusion. In addition, the instructor should be able to teach those basic courses and should be available at those basic courses time slot. Since this model is not complicated, we will use linear programming for solving the model in this stage. These 24 basic course sections consist of 2 sections of 2301101, 3 sections of 2301103, 7 sections of 2301107, 2 sections of 2301113, 2 sections of 2301115, 4 sections of 2301117, 3 sections of 2301119, and 1 section of 2301675. We start with explanations of notations, parameters, decision variables. And then, we present objective function and constraints for the second stage model.

4.2.1 Notations and parameters

Let I be the set of all instructors,

J be the set of all courses,

J_i be the set of courses that instructor i can teach,

K_j be the set of sections of course j,

T be the set of time slots,

T_i be the set of all available time slots of instructor i,

o_i be the total workload of instructor i, readjusted from the first stage model,

r_i be the requested workload of instructor i,

w_j be the workload of course j,

and $p_{i,j}$ be the teaching preference value of instructor i who prefers to teach course j.

4.2.2 Decision variables

Let $x_{i,j,k,t}$ be a binary variable taking value 1 if instructor i teaches course j section k at time slot t, otherwise it equals 0,

$x_{i,j,k}$ be a binary variable taking value 1 if instructor i teaches course j section k, otherwise it equals 0,

s_i be a positive part of the subtraction between the assigned and requested workload for instructor i,

and u_i be a negative part of the subtraction between the assigned and requested workload for instructor i.

4.2.3 Objective function

The objective function in this stage is the same as the first stage model because we have the same goal of balancing workload and maximizing preferences

of instructors.

$$\min -\alpha_1 f_1 + \alpha_2 f_2 \tag{4.31}$$

where $f_1 = \sum_{i \in I} \sum_{j \in J_i} p_{i,j} x_{i,j}$ is the total teaching preferences,

$f_2 = \sum_{i \in I} \left| (\sum_{j \in J} w_j x_{i,j} + o_i) - r_i \right| = \sum_{i \in I} (s_i + u_i)$ is the total difference between instructors requested workload and their assigned workload such that α_1, α_2 are the weight.

4.2.4 Constraints

1. Instructors would not assign courses that they cannot teach.

$$x_{i,j,k,t} = 0, \quad \forall i \in I, \forall j \in J \setminus J_i, \forall k \in K_j, \forall t \in T \tag{4.32}$$

2. Each instructor can only teach at most one section for each course.

$$\sum_{k \in K_j} x_{i,j,k} \leq 1, \quad \forall i \in I, \forall j \in J \tag{4.33}$$

3. Instructor i is allowed to teach at most one course section at any time slot.

$$\sum_{j \in J} \sum_{k \in K_j} x_{i,j,k,t} \leq 1, \quad \forall i \in I, \forall t \in T \tag{4.34}$$

4. Instructor i is available only in the set of time slot T_i.

$$\sum_{j \in J} \sum_{k \in K_j} x_{i,j,k,t} = 0, \quad \forall i \in I, \forall t \notin T_i \tag{4.35}$$

5. Each course section can be taught by only one instructor.

$$\sum_{i \in I} x_{i,j,k} = 1, \quad \forall j \in J, \forall k \in K_j \tag{4.36}$$

6. These constraints will force at least one of s_i and u_i to be zero.

$$s_i \geq (\sum_{j \in J} w_j x_{i,j} + o_i) - r_i, \quad \forall i \in I \tag{4.37}$$

$$u_i \geq r_i - (\sum_{j \in J} w_j x_{i,j} + o_i), \quad \forall i \in I \tag{4.38}$$

7. Instructor i must be assigned to course j first before be able to know course section k.

$$x_{i,j} \geq x_{i,j,k}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j \tag{4.39}$$

8. Instructor i must be assigned to section k of course j before choosing time slot t.

$$x_{i,j,k} \geq x_{i,j,k,t}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j, \forall t \in T \tag{4.40}$$

9. All variables are binary variables.

$$x_{i,j,k,t},\ x_{i,j,k},\ x_{i,j} \in \{0,1\}, \quad \forall i \in I, \forall j \in J, \forall k \in K_j, \forall t \in T \tag{4.41}$$

10. The positive part (s_i) and the negative part (u_i) are the subtraction between the assigned and requested workload for each instructor i must be non-negative.

$$s_i,\ u_i \geq 0, \quad \forall i \in I \tag{4.42}$$

We solve the two-stage model in CPLEX Studio IDE 12.8.0. The code of the first stage model (file) and the second stage model (file) can be found in Appendix L (Appendix J) and Appendix M (Appendix K), respectively. The results of our model are provided in the next chapter.

CHAPTER V

MAIN RESULTS AND CONCLUSIONS

This chapter presents the result from our model by showing a full timetable and comparing the results with the real data of our department timetable in the first semester 2019 and the first stage model alone. At the end, we will conclude this thesis.

5.1 Main results

The final outcome of some instructors of our two-stage model is presented in Table 5.1. The full timetable shows who teach which course section, when, where, and what student groups they teach. We also present the full timetable in terms of the teaching timetable for each instructor in Appendix G and for each student group in Appendix H. The full timetable of the classroom schedule for each classroom is in Appendix I.

After getting the full timetable, we would like to show that our two-stage model is better in both aspects: balancing among all assigned workload and maximizing the total teaching preferences of instructors. From Section 3.2, we described about the requested workload for each instructor. If the instructor receives the assigned workload more than their requested workload or overload, it may directly affect the teaching and research efficiency. On the other hand, if the instructor receives the assigned workload less than their requested workload or underload, it is unfair to other instructors. Therefore, balancing workload is to make the assigned workload as close as possible the requested workload, in other word, the difference of assigned and requested workload should be as close to zero as possible. Hence, we want to reduce the number of instructors whose the assigned workload differs

		Instructors				
	Time slots	A1	A9	A15	...	A20
Monday	1		2301170 (4) MHMK-207 Science-1	2301103 (1) d2 Pharm Sci	...	2301107 (3) d6 Engineering-1
	⋮	⋮	⋮	⋮	⋮	⋮
	16	2301386 (1) TAB-230 Math-3			...	2301181 (1) MHMK-208 Math-1
Tuesday	17				...	2301119 (3) d5 Account
	⋮	⋮	⋮	⋮	⋮	⋮
	32				...	
Wednesday	17				...	2301119 (3) d5 Account
	⋮	⋮	⋮	⋮	⋮	⋮
	48				...	
Thursday	17				...	2301181 (1) TAB-220 Math-1
	⋮	⋮	⋮	⋮	⋮	⋮
	64				...	
Friday	65		2301170 (4) TAB-221 Science-1		...	
	⋮	⋮	⋮	⋮	⋮	⋮
	80				...	

Table 5.1: The full timetable using the two-stage model.

from the requested workload. The difference of requested and the assigned workload are shown in Table 5.2. The difference is set up by using the range of 2 units. We can see that the number of instructors in the first range in the department timetable seems to be greater than the other models. However, the number of instructors whose differences are up to 4 units of workload obtained by our two-stage model is greater than both the department timetable and the first stage model alone. Moreover, the number of instructors who have the differences of assigned

Difference range (units of workload)	The number of instructors		
	Manual (Department)	First stage model alone	Two-stage model
0–2	16	13	14
2–4	10	14	18
4–6	8	7	10
6–8	4	8	2
8–10	9	2	8
10–12	0	5	3
more than 12	13	11	5
$\sum_{i \in I}(s_i + u_i)$	462.64	396.62	330.82

Table 5.2: Comparison on the difference of the requested and the assigned workload obtained by the department timetable, the first stage model alone, and the two-stage model.

workload and requested workload exceeding 12 units of two-stage model is only 5 instructors. Since the other models in more than 12 units of difference range have 11 and 13 instructors, our two-stage model is the best among all other models. In addition, the total difference between the assigned and requested workload of all instructors ($\sum_{i \in I}(s_i + u_i)$) in our two-stage model is less than the other models too.

On the other objective, we would like to match instructors with the most of their preferred courses. The number of assigned courses in each preference rank is presented in Table 5.3.

Preference rank	The number of assigned courses				
	Manual (Department)		First stage model alone	Two-stage model	
	Before midterm	After midterm		Before midterm	After midterm
1	83	83	73	75	81
2	13	14	24	24	22
3	3	3	7	4	4
4	2	3	4	6	4
5	21	19	14	13	11

Table 5.3: The number of assigned courses in each preference rank.

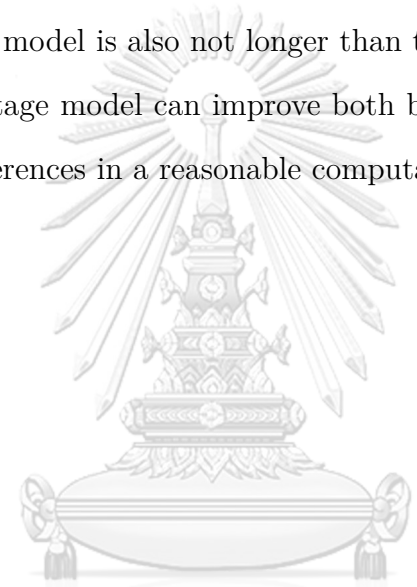
The total number of the first two ranks of assigned courses in our two-stage model is 99 course sections before the midterm and 103 course sections after the midterm which is greater than the ones obtained by the department and the first stage model alone. In the similar fashion, the number of non-preferred course sections in two-stage model is only 11 course sections before midterm part and 13 course sections after midterm part which is less than both the department and the first stage model alone. Moreover, the department has over 19 course sections in non-preferred course sections. This result guarantees that the two-stage model can match instructors and courses effectively.

Solving the second stage model only takes 10 seconds to reach an optimal solution using a laptop with an Intel Core i7 CPU and 8 GB of RAM. However, our two-stage model contains 1761 variables and 73695 constraints in the first stage model. Therefore, we limit the run time of the first stage of our two-stage model to be 8000 seconds ($\sim$ 133 minutes) using a computer with an Intel Xeon Platinum CPU and 127 GB of RAM. It is reasonable time limit according to other research references such as [18] whose run time limit is 10800 seconds (180 minutes).

5.2 Conclusions

This thesis deals with assigning instructors to their preferred courses by balancing instructor workload and maximizing preferences while satisfying all restrictions of Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University. Our timetable also schedules courses to suitable classrooms at appropriate time slots for each student group. The two-stage model is applied in this thesis by dividing some basic courses into two parts: before the midterm and after the midterm. This idea helps sharing the other half of the teaching workload to other instructors who teach after midterm part. The result in Table 5.2 shows that our two-stage model can reduce the overall instructors

workload. A number of instructors who have the difference range of the assigned workload and the requested workload that more than 12 units in our two-stage model is fewer than the department's assignment. On the other aims, the result in Table 5.3 also shows that our two-stage model can reduce the number of course sections in rank 5 or non-preferred courses in the both before and after midterm parts. Moreover, there are also a lot of varieties and specific constraints since our model has constraints that correspond to the department restrictions. The number of constraints in our model is more than the other research work while the limit run time in our model is also not longer than the other work. These results verify that our two-stage model can improve both balancing instructor workload and maximizing preferences in a reasonable computation time.



REFERENCES

[1] S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan. A hybrid metaheuristic approach to the university course timetabling problem. *Journal of Heuristic*, 18(1):1–23, 2012.

[2] H. Cambazard, E. Hébrard, B. O'Sullivan, and P. Alexandre. Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals of Operations Research*, 194(1):111–135, 2012.

[3] B. Domenech and A. Lusa. A MILP model for the teacher assignment problem considering teachers's preferences. *European Journal of Operational Research*, 249(3):1153–1160, 2016.

[4] B. Genc and B. O'Sullivan. A two-phase constraint programming model for examination timetabling at university college cork. In *Principles and Practice of Constraint Programming 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7–11, 2020, Proceedings*, pages 724–742. Springer, 2020.

[5] A. Gorka and P. Thipwiwatpotjana. The importance of fuzzy preference in course assignment problem. *Mathematical Problems in Engineering*, 2015(1): 1–8, 2015.

[6] A. A. Gozali, B. Kurniawan, W. Weng, and S. Fujimura. Solving university course timetabling problem using localized island model genertic algoruthm with dual dynamic migratin policy. *IEEJ transactions on electrical and electronic engineering*, 15:389–400, 2019.

[7] A. Gunawan, K. M. Ng, and K. L. Poh. A hybridized lagrangian relaxation and simulated annealing method for the course timetabling problem. *Computers Operations Research*, 39:3074–3088, 2012.

[8] S. I. Hossain, M. A. H. Akhand, M. I. R. Shuvo, and N. Siddique. Optimization of university course scheduling problem using particle swarm optimization with selective search. *Expert System with Applications*, 127:9–24, 2019.

[9] K. Y. Junn, J. H. Obit, and R. Alfred. A constraint programming approach to solving university course timetabling problem (UCTP). In *Advanced Science Letters*, volume 4, pages 400–407, 2016.

[10] K. Y. Junn, J. H. Obit, and R. Alfred. The study of genetic algorithm approach to solving university course timetabling problem. In *Computational Science and Technology*, pages 454–463. Springer, Singapore, 2017.

[11] A. R. Komijan and M. N. Koupael. A mathematical model for university course scheduling; a case study. *International Journal of Technical Research and Applications*, 19:20–25, 2015.

[12] B. Naderi. Modeling and scheduling university course timetabling problems. In *International Journal of Research in Industrial Engineering*, volume 5, pages 1–15, 2016.

[13] O. A. Odeniyi, E. O. Omidiora, S. O. Olabiyisi, and C. A. Oyeleye. A mathematical programming model and enhanced simulated annealing algorithm for the school timetabling problem. *Asian Journal of Research in Computer Science*, 5(3):21–38, 2020.

[14] P. Thipwiwatpotjana. Course assignment problem with interval requested workload. In *2014 IEEE Conference on Norbert Wiener in the 21st Century*

(21CW). IEEE, 2014.

[15] Christos Valouxis and Efthymios Housos. Constraint programming approach for school timetabling. *Computers Operations Research*, 30(10):1555–1572, 2003.

[16] R. Visuthirattanamanee and P. Thipwiwatpotjana. Teaching course assignment problem with instructors fuzzy satisfaction. In *Proceedings of The 19th Annual Meeting in Mathematics (AMM)*, 2014.

[17] T. Wanga, N.Meskensb, and D. Duvivier. Scheduling operating theatres: Mixed integer programming vs. constraint programming. *European Journal of Operational Research*, 247:401–413, 2015.

[18] P. Yasari, M. Ranjbar, N. Jamili, and M. H. Shaelaie. A two-stage stochastic programming approach for a multi-objective course timetabling problem with courses cancelation risk. *Computers and Industrial Engineering*, 130:650–660, 2019.

APPENDIX

APPENDIX A: A survey of preference ranking of each course.

(The data will be expired in December 2023.)

Figure 1: QR code of a survey of preference ranking of each course.

URL: https://qrgo.page.link/KNAqq

APPENDIX B: The adjusted teaching preference.

(The data will be expired in December 2023.)

Figure 2: QR code of the adjusted teaching preference.

URL: https://qrgo.page.link/CFJU8

APPENDIX C: The number of maximum teaching hour per day, requested, seminar, and advisor workloads of instructors.

(The data will be expired in December 2023.)

Figure 3: QR code of the number of maximum teaching hour per day, requested, seminar, and advisor workloads of instructors.

URL: https://qrgo.page.link/hPcK5

APPENDIX D: The number of student, workload, and credit for each course.

(The data will be expired in December 2023.)

Figure 4: QR code of the number of student, workload, and credit for each course.

URL: https://qrgo.page.link/7Johc

APPENDIX E: Classrooms and their capacities.

(The data will be expired in December 2023.)

Figure 5: Classrooms and their capacities.

URL: https://qrgo.page.link/QegEs

APPENDIX F: Required duration and its repetition of all courses.

(The data will be expired in December 2023.)

Figure 6: Required duration and its repetition of all courses.

URL: https://qrgo.page.link/pB3mb

APPENDIX G: The full timetable for each instructor.

(The data will be expired in December 2023.)

Figure 7: QR code of the full timetable for each instructor.

URL: https://qrgo.page.link/6nh4P

APPENDIX H: The full timetable for each student group.

(The data will be expired in December 2023.)

Figure 8: QR code of the full timetable for each student group.

URL: https://qrgo.page.link/PeVvB

APPENDIX I: The full timetable for each classroom.

(The data will be expired in December 2023.)

Figure 9: QR code of the full timetable for each classroom.

URL: https://qrgo.page.link/6JTVv

APPENDIX J: The code for the first stage model file.

(The data will be expired in December 2023.)

Figure 10: QR code of the code for the first stage model file.

URL: https://qrgo.page.link/8YHAp

APPENDIX K: The code for the second stage model file.

(The data will be expired in December 2023.)

Figure 11: QR code of the code for the second stage model file.

URL: https://qrgo.page.link/C5wiy

APPENDIX L: The code for the first stage model.

The first stage model.mod


```
using CP;
execute{
}
tuple Pair {
string a;
string b;
};
tuple td {
string teacher;
string discipline;
int preference;
```

```
12  };
13  tuple Requirement {
14  string class; // a set of pupils
15  string discipline; // (course) what will be taught
16  string section;
17  int Duration; // course duration
18  int repetition; // how many time the course is repeated
19  };
20  tuple si {
21  string Room;
22  int Capacity;
23  };
24  tuple sii {
25  string discipline;
26  int numberstudent;
27  int disciplineworkload;
28  };
29  tuple siiii {
30  string teacher;
31  int maxteachinghourperday;
32  int weightedrequestedworkload;
33  int weightedseminarworkload;
34  int weightedadvisorworkload;
35  };
36
37  //
38  // user given model data
39  //
40  {Pair} NeedBreak = ...; // courses that should not be contiguous in
        time
41  {string} MorningDiscipline = ...; // courses that must be taught in
        the morning
```

```
42  {string} AfternoonDiscipline = ...; // courses that must be taught
        in the afternoon
43  {string} SpecificDiscipline=...;
44  {td} TeacherDisciplineSet = ...; // what are the instructor skills
45  {Pair} DedicatedRoomSet = ...; // a set of courses requiring special
        rooms
46  {Requirement} RequirementSet = ...; // the educational program
47  {string} LabRoom = ...;      // the set of available laboratory
48  {string} SmallRoom = ...;     // the set of available room for no
        more than 25 students
49  {string} MediumRoom = ...;     // the set of available room for no
        more than 65 students
50  {string} LargeRoom = ...;     // the set of available room for no
        more than 300 students
51  {string} DummyRoom = ...;     // the set of non faculty of Science
        room
52  int BreakDuration = ...; // time interval between two courses
53  int DayDuration = ...; // must be even (morning duration equals
        afternoon duration)
54  int MeetingDay = ...;     // Wednesday afternoon
55  int Day = ...; // how many worked days per period
56  {si} RoomCapacitySet = ...;
57  {sii} DisciplineSet = ...;
58  {siiii} WorkloadSet = ...;
59
60  //
61  // vocabularies
62  //
63  {string} Class = {c | <c,d,s,u,n> in RequirementSet };
64  {string} SciGroup = {c | <c,d,s,u,n> in RequirementSet:
65  c in {"AMCS","Pure-1","Pure-2","Comp M.Sc",
66  "Math-4","Math-3","Math-2","Math-1",
67  "Comp-4","Comp-3","Comp-2","Comp-1",
```

```
68      "Science-3","Science-2","Science-1"}};
69      {string} NotSciGroup = {c | <c,d,s,u,n> in RequirementSet: c in
            Class && c not in SciGroup};
70      {string} Teacher = { t | <t,d,p> in TeacherDisciplineSet };
71      {string} Discipline = {d | <c,d,s,u,n> in RequirementSet };
72      {string} Section = {s | <c,d,s,u,n> in RequirementSet };
73      {string} SciRoom = LabRoom union SmallRoom union MediumRoom union
            LargeRoom;
74      {string} Room = SciRoom union DummyRoom;
75
76      //
77      // time expressions
78      //
79      int HalfDayDuration = DayDuration div 2;
80      int MaxTime = DayDuration*Day;
81      range Time = 0..MaxTime-1;
82      range WedAfternoonPeriod = 1..HalfDayDuration;
83      range DayID = 1..Day;
84
85      //
86      // convenience expressions for room compatibility
87      //
88      int StudentPerDiscipline[d in Discipline] = (card({m | <d,m,n> in
            DisciplineSet})==0)?0:first({m | <d,m,n> in DisciplineSet});
89      int RoomCapacity[x in Room] = (card({n | <x,n> in RoomCapacitySet})
            ==0)?0:first({n | <x,n> in RoomCapacitySet});
90      int PossibleRoom[d in Discipline, c in Class, x in Room] =
91      <d,x> in DedicatedRoomSet
92      || (0 == card({<k,z> | k in Discipline, z in Room
93      : (<k,x> in DedicatedRoomSet) || (<d,z> in DedicatedRoomSet)})
94      && (StudentPerDiscipline[d] <= RoomCapacity[x])
95      && (StudentPerDiscipline[d] > 40 || x not in LargeRoom)
```

```
96    && ((c in SciGroup && x in SciRoom) || (c in NotSciGroup && x in
          DummyRoom)));
97
98    int NbRoom = card(Room);
99    range RoomId = 0..NbRoom-1;
100
101   {int} PossibleRoomIds[d in Discipline,c in Class] =
102   {i | i in RoomId, x in Room
103   : (PossibleRoom[d,c,x] == 1) && (i == ord(Room,x))};
104
105   //
106   // convenience expressions for instructor skills
107   //
108   {string} PossibleTeacherDiscipline[x in Teacher] = {d | <x,d,p> in
          TeacherDisciplineSet};
109   int NbTeacher = card(Teacher);
110   range TeacherId = 0..NbTeacher-1;
111
112
113   //
114   // convenience expressions for instructor workloads
115   //
116   int MaxTeachingHourPerDay[x in Teacher] = (card({m | <x,m,n,p,q> in
          WorkloadSet })==0)?0:first({m | <x,m,n,p,q> in WorkloadSet});
117   int RequestedWorkload[x in Teacher] = (card({n | <x,m,n,p,q> in
          WorkloadSet})==0)?0:first({n | <x,m,n,p,q> in WorkloadSet});
118   int SeminarWorkload[x in Teacher] = (card({p | <x,m,n,p,q> in
          WorkloadSet})==0)?0:first({p | <x,m,n,p,q> in WorkloadSet});
119   int AdvisorWorkload[x in Teacher] = (card({q | <x,m,n,p,q> in
          WorkloadSet})==0)?0:first({q | <x,m,n,p,q> in WorkloadSet});
120
121   {int} PossibleTeacherIds[d in Discipline] =
122   {i | i in TeacherId, z in Teacher
```

```
123     : i == ord(Teacher, z)
124     && d in PossibleTeacherDiscipline[z] };
125
126     //
127     // convenience expressions for requirement instantiation
128     //
129     // for a given requirement, an instance is one course occurrence
130     tuple Instance {
131     string class;
132     string discipline;
133     string section;
134     int Duration;
135     int repetition;
136     int id;
137     int requirementId;
138     };
139     {Instance} InstanceSet = {
140     <c,d,s,u,n,i,z> | <c,d,s,u,n> in RequirementSet
141     , z in ord(RequirementSet,<c,d,s,u,n>) .. ord(RequirementSet,<c,d,s,
            u,n>)
142     , i in 1..n
143     };
144
145     //
146     // decision variables
147     //
148     dvar int Start[InstanceSet] in Time; // the course starting point
149     dvar int room[InstanceSet] in RoomId; // the room in which the
            course is held
150     dvar int teacher[InstanceSet] in TeacherId; // the instructor in
            charge of the course
151     dvar int TotalWorkload[Teacher];
152     dvar int TeachingWorkload[Teacher];
```

```
153     dvar int TeachingPreference[Teacher];
154     dvar int PenaltyWorkload[Teacher];
155     dvar int NumTeachingDiscipline[Teacher];
156     dvar int TeachingHourPerDay[Teacher][DayID];
157
158     //
159     // helper variables
160     //
161     dvar int End[InstanceSet] in Time; // the course end time
162     dvar int classTeacher[Class,Discipline,Section] in TeacherId; //
            teacher working once per time point
163     dvar int makespan in Time;       // ending date of last course
164     dvar int SumAbsPenaltyWorkload;
165     dvar int SumTeachingPreference;
166
167     //
168     // search setup
169     //
170     execute {
171     writeln("MaxTime = ", MaxTime);
172     writeln("DayDuration = ", DayDuration);
173     writeln("Teacher = ", Teacher);
174     writeln("Discipline = ", Discipline);
175     writeln("Class = ", Class);
176     var f = cp.factory;
177     var selectVar = f.selectSmallest(f.domainSize());
178     var selectValue = f.selectRandomValue();
179     var assignRoom = f.searchPhase(room, selectVar, selectValue);
180     var assignTeacher = f.searchPhase(teacher, selectVar, selectValue);
181     var assignStart = f.searchPhase(Start, selectVar, selectValue);
182     cp.setSearchPhases(assignTeacher, assignStart, assignRoom);
183     var p = cp.param;
184     //p.logPeriod = 10000;
```

```
185     //p.searchType = "DepthFirst";
186     //p.searchType = "Multipoint";
187     //p.searchType = "Restart";
188     p.timeLimit = 10000; // number in seconds
189     //p.FailLimit = 10000;
190     }
191
192     minimize SumAbsPenaltyWorkload - SumTeachingPreference;
193
194     subject to {
195     SumAbsPenaltyWorkload == sum(x in Teacher) abs(PenaltyWorkload[x]);
196     SumTeachingPreference == sum(x in Teacher) TeachingPreference[x];
197
198     // 1) help proving optimality
199     makespan == max(r in InstanceSet) End[r];
200     // 2) ensure the discipline ends after it starts
201     forall(r in InstanceSet)
202     End[r] == r.Duration + Start[r];
203     // 3) ensure course numeration is chronological
204     forall(i, j in InstanceSet
205     : i.id < j.id
206     && i.requirementId == j.requirementId)
207     Start[i] < Start[j];
208     // 4) ensure that an instructor is required once at any time slot.
209     forall(r in InstanceSet, x in Teacher) {
210     if(r.discipline in PossibleTeacherDiscipline[x])
211     (sum(o in InstanceSet: o.discipline in PossibleTeacherDiscipline[x])
212     (Start[o] >= Start[r])
213     *(Start[o] < End[r])
214     *(teacher[o] == ord(Teacher,x))) < 2;
215     }
216     // 5) ensure the instructor can teach the course
217     forall(r in InstanceSet)
```

```
218     teacher[r] in PossibleTeacherIds[r.discipline];
219     // 6) ensure that a classroom is required once at any time slot.
220     forall(r in InstanceSet, x in Room) {
221     if(PossibleRoom[r.discipline,r.class,x] == 1)
222     (sum(o in InstanceSet : 1 == PossibleRoom[o.discipline,o.class,x])
223     (Start[o] >= Start[r])
224     *(Start[o] < End[r])
225     *(room[o] == ord(Room,x))) < 2;
226     }
227     // 7) ensure the classroom can support the course
228     forall(r in InstanceSet)
229     room[r] in (PossibleRoomIds[r.discipline,r.class]);
230     // 8) ensure that a class follows one course at a time (the
          different section in the same course can be taught at the same
          time)
231     forall(r in InstanceSet, x in Class) {
232     if(r.class == x)
233     (sum(o in InstanceSet: o.class == x && r.discipline != o.discipline)
234     (Start[o] >= Start[r])*(Start[o] < End[r])) == 0;
235     }
236     // 9) ensure that for given class and course (but the different
          section), the instructor is always the same
237     forall(c in Class, d in Discipline, s in Section, r in InstanceSet
238     : r.class == c && r.discipline == d && r.section == s)
239     teacher[r] == classTeacher[c,d,s];
240     // 10) ensure a course starts and end the same half-day and the same
           day
241     forall(i in InstanceSet)
242     (Start[i] div HalfDayDuration) == ((End[i]-1) div HalfDayDuration);
243     // 11) insert break duration between specified courses
244     forall(ordered i, j in InstanceSet, a,b in Discipline
245     : (<b,a> in NeedBreak || <a,b> in NeedBreak)
246     && i != j
```

```
247     && i.class == j.class
248     && ((i.discipline == a && j.discipline == b)
249     || (i.discipline == b && j.discipline == a)))
250     // 12) courses do not belong to the same day
251     ((Start[i] div DayDuration) != (Start[j] div DayDuration)) ||
252     // 13) courses do not belong to the same half-day
253     ((Start[i] div HalfDayDuration) != (Start[j] div HalfDayDuration))
            ||
254     // 14) courses are separated by BreakDuration
255     ((Start[i] > End[j])*(Start[i] - End[j]) +
256     (Start[j] > End[i])*(Start[j] - End[i])) >= BreakDuration;
257     // 15) no classes on every Wednesday afternoon
258     forall(i in InstanceSet, d in Discipline : i.discipline == d)
259     sum(w in WedAfternoonPeriod) (Start[i] == MeetingDay*DayDuration - w
            ) <= 0;
260     // 16) ensure that the morning courses end in the morning
261     forall(d in MorningDiscipline, i in InstanceSet : i.discipline == d)
262     (Start[i] mod DayDuration) < HalfDayDuration;
263     // 16.1) ensure that the afternoon courses end in the afternoon
264     forall(d in AfternoonDiscipline, i in InstanceSet : i.discipline ==
            d)
265     (Start[i] mod DayDuration) >= HalfDayDuration;
266     //
267     // 17) avoid the course taught consecutive day
268     //
269     // 17.1) avoid the course that taught 2 and 3 times a week teaching
            consecutive days
270     forall(ordered i,j in InstanceSet: i.discipline == j.discipline && i
            .section == j.section && i.class == j.class
271     && i.repetition >= 2 && i.repetition <= 3 && j.repetition >= 2 && j.
            repetition <=3)
272     ((Start[j] div DayDuration) - (Start[i] div DayDuration)) >= 2;
```

```
273   // 17.2) avoid the course that have the different duration, teaching
          consecutive days
274   forall(ordered i,j in InstanceSet: i.discipline == j.discipline && i
          .section == j.section && i.class == j.class
275   && i.Duration != j.Duration && i.repetition == 1 && j.repetition ==
          1)
276   ((Start[j] div DayDuration) - (Start[i] div DayDuration)) >= 2;
277   // 17.3) avoid the same course and the same section teach at the
          same day
278   forall(ordered i,j in InstanceSet: i.discipline == j.discipline && i
          .section == j.section && i.class == j.class)
279   ((Start[j] div DayDuration) != (Start[i] div DayDuration));
280   // 17.4) avoid the same course and the same section but different
          duration teach at the same day
281   forall(i,j in InstanceSet: i.discipline == j.discipline && i.section
          == j.section && i.class == j.class
282   && i.Duration != j.Duration)
283   ((Start[j] div DayDuration) != (Start[i] div DayDuration));
284   // 18) there are the maximum number of lectures for assigned to the
          instructors in a working day.
285   forall(x in Teacher, i in DayID)
286   TeachingHourPerDay[x][i] == (sum(r in InstanceSet: r.discipline in
          PossibleTeacherDiscipline[x])
287   ((i-1)*DayDuration <= End[r] < i*DayDuration)*r.Duration*(teacher[r]
          == ord(Teacher,x)));
288   forall(x in Teacher, i in DayID)
289   TeachingHourPerDay[x][i] <= MaxTeachingHourPerDay[x];
290   // 19) calculating the teaching workload assigned to the instructors
291   forall(x in Teacher)
292   TeachingWorkload[x] ==
293   sum(r in InstanceSet, s in DisciplineSet: r.discipline in
          PossibleTeacherDiscipline[x]
```

```
294     && s.discipline in PossibleTeacherDiscipline[x] && r.discipline in
            SpecificDiscipline
295     && s.discipline in SpecificDiscipline && r.discipline == s.
            discipline)
296     s.disciplineworkload/(sum(p in InstanceSet: p.discipline == r.
            discipline && p.section == r.section
297     && p.class == r.class) r.repetition/r.repetition)/2*(teacher[r] ==
            ord(Teacher,x)) +
298     sum(r in InstanceSet, s in DisciplineSet: r.discipline in
            PossibleTeacherDiscipline[x]
299     && s.discipline in PossibleTeacherDiscipline[x] && r.discipline not
            in SpecificDiscipline
300     && s.discipline not in SpecificDiscipline && r.discipline == s.
            discipline)
301     s.disciplineworkload/(sum(p in InstanceSet: p.discipline == r.
            discipline && p.section == r.section
302     && p.class == r.class) r.repetition/r.repetition)*(teacher[r] == ord
            (Teacher,x));
303     // 20) calculating the total workload assigned to the instructors
304     forall(x in Teacher)
305     TotalWorkload[x] == SeminarWorkload[x] + AdvisorWorkload[x] +
            TeachingWorkload[x];
306     // 21) calculating the difference workload between the total
            workload and the requedted workload for each instructor
307     forall(x in Teacher)
308     PenaltyWorkload[x] == TotalWorkload[x] - RequestedWorkload[x];
309     // 22) calculating the preference course assigned to the instructors
310     forall(x in Teacher)
311     TeachingPreference[x] == sum(r in InstanceSet, d in
            TeacherDisciplineSet: d.discipline in PossibleTeacherDiscipline[
            x] &&
312     r.discipline in PossibleTeacherDiscipline[x] && r.discipline == d.
            discipline && d.teacher == x)
```

```
313     d.preference/(sum(p in InstanceSet: p.discipline == r.discipline &&
            p.section == r.section
314     && p.class == r.class) r.repetition/r.repetition)*(teacher[r] == ord
            (Teacher,x));
315     // 23) Each instructor can only teach at most one section for each
            course.
316     forall(x in Teacher, i,j in InstanceSet: i.discipline == j.
            discipline && i.section != j.section)
317     (teacher[i] == ord(Teacher,x))*(teacher[j] == ord(Teacher,x)) < 1;
318     // 24) Each course should start at the begining of an hour
319     forall(i in InstanceSet, d in Discipline: i.discipline == d)
320     (Start[i] mod 2 == 0);
321     // 25) limit 3 courses for each instructor
322     forall(x in Teacher)
323     NumTeachingDiscipline[x] == sum(r in InstanceSet, d in
            TeacherDisciplineSet: d.discipline in PossibleTeacherDiscipline[
            x] &&
324     r.discipline in PossibleTeacherDiscipline[x] && r.discipline == d.
            discipline && d.teacher == x)
325     d.preference/(sum(p in InstanceSet: p.discipline == r.discipline &&
            p.section == r.section
326     && p.class == r.class) r.repetition/r.repetition)/d.preference*(
            teacher[r] == ord(Teacher,x));
327
328     forall(x in Teacher)
329     NumTeachingDiscipline[x] <= 3;
330     };
331
332     //
333     // generate time table
334     //
335     tuple Course {
336     int Time;
```

```
337     string class;
338     string teacher;
339     string discipline;
340     string section;
341     string room;
342     int id;
343     int repetition;
344     };
345     tuple T {
346     string discipline;
347     string section;
348     int id;
349     int repetition;
350     string room;
351     string class;
352     };
353     tuple R {
354     string discipline;
355     string section;
356     int id;
357     int repetition;
358     string teacher;
359     string class;
360     };
361     tuple D {
362     string section;
363     string teacher;
364     string room;
365     string class;
366     };
367     tuple dis {
368     string discipline;
369     string section;
```

```
370     };
371     tuple s {
372     string discipline;
373     string section;
374     };
375
376     {Course} timetable[t in Time][c in Class] = {
377     <t,c,p,d,s,r,i,n>
378     | d in Discipline
379     , s in Section
380     , r in Room
381     , x in InstanceSet
382     , n in x.repetition..x.repetition
383     , p in Teacher
384     , i in x.id..x.id
385     : (t >= Start[x])
386     && (t < End[x])
387     && (x.class == c)
388     && (room[x] == ord(Room,r))
389     && (teacher[x]==ord(Teacher,p))
390     && (d == x.discipline)
391     && (s == x.section)
392     };
393
394     {T} teachertimetable[t in Time][p in Teacher] = {
395     <d,s,i,n,r,c>
396     | d in Discipline
397     , s in Section
398     , x in InstanceSet
399     , i in x.id..x.id
400     , n in x.repetition..x.repetition
401     , r in Room
402     , c in Class
```

```
403     : (t >= Start[x])
404     && (t < End[x])
405     && (ord(Teacher,p) == teacher[x])
406     && (d == x.discipline)
407     && (s == x.section)
408     && (room[x] == ord(Room,r))
409     && (x.class == c)
410     };
411
412     {R} roomtimetable[t in Time][r in Room] = {
413     <d,s,i,n,p,c>
414     | d in Discipline
415     , s in Section
416     , x in InstanceSet
417     , i in x.id..x.id
418     , n in x.repetition..x.repetition
419     , p in Teacher
420     , c in Class
421     : (t >= Start[x])
422     && (t < End[x])
423     && (ord(Teacher,p) == teacher[x])
424     && (d == x.discipline)
425     && (s == x.section)
426     && (room[x] == ord(Room,r))
427     && (x.class == c)
428     };
429
430     {D} disciplinetimetable[t in Time][d in Discipline] = {
431     <s,p,r,c>
432     | p in Teacher
433     , s in Section
434     , r in Room
435     , c in Class
```

```
436     , x in InstanceSet
437     : (t >= Start[x])
438     && (t < End[x])
439     && (ord(Teacher,p) == teacher[x])
440     && (d == x.discipline)
441     && (s == x.section)
442     && (room[x] == ord(Room,r))
443     && (x.class == c)
444     };
445
446     // force execution of postprocessing expressions
447     execute POST_PROCESS {
448     timetable;
449     for(var c in Class) {
450     writeln("Class ", c);
451     var day = 0;
452     for(var t = 0; t < makespan; t++) {
453     if(t % DayDuration == 0) {
454     day++;
455     writeln("Day ", day);
456     }
457     if(t % DayDuration == HalfDayDuration)
458     writeln("Lunch break");
459
460     if(t >= MeetingDay*DayDuration-HalfDayDuration && t <= MeetingDay*
            DayDuration-1)
461     writeln((t % DayDuration)+1, "\tDepartment Meeting");
462
463     var activity = 0;
464     for(var x in timetable[t][c]) {
465     activity++;
466     writeln((t % DayDuration)+1, "\t",
467     x.room, "\t",
```

```
468     x.discipline,
469     "(", x.section, ")", "\t",
470     x.id, "/",
471     x.repetition, "\t",
472     x.teacher);
473     }
474     if(activity == 0 && (t < MeetingDay*DayDuration-HalfDayDuration || t
            > MeetingDay*DayDuration-1))
475     writeln((t % DayDuration)+1, "\tFree time");
476     }
477     }
478     }
479
480     execute POST_PROCESS1 {
481     teachertimetable;
482     for(var r in Teacher) {
483     writeln("Teacher ", r);
484     var day = 0;
485     for(var t = 0; t < makespan; t++) {
486     if(t % DayDuration == 0) {
487     day++;
488     writeln("Day ", day);
489     }
490
491     var activity = 0;
492     for(var x in teachertimetable[t][r]) {
493     activity++;
494     writeln((t % DayDuration)+1, "\t",
495     x.discipline,
496     "(", x.section, ")", "\t",
497     x.id, "/",
498     x.repetition, "\t",
499     x.room);
```

```
500     }
501     }
502     }
503     }
504
505     execute POST_PROCESS2 {
506     roomtimetable;
507     for(var r in Room) {
508     writeln("Room ", r);
509     var day = 0;
510     for(var t = 0; t < makespan; t++) {
511     if(t % DayDuration == 0) {
512     day++;
513     writeln("Day ", day);
514     }
515
516     var activity = 0;
517     for(var x in roomtimetable[t][r]) {
518     activity++;
519     writeln((t % DayDuration)+1, "\t",
520     x.discipline,
521     "(", x.section, ")", "\t",
522     x.id, "/",
523     x.repetition, "\t",
524     x.teacher);
525     }
526     }
527     }
528     }
529
530     execute POST_PROCESS3 {
531     disciplinetimetable;
532     writeln("DisciplineTimetable");
```

```
533     for(var d in Discipline) {
534     var day = 0;
535     for(var t = 0; t < makespan; t++) {
536     if(t % DayDuration == 0) {
537     day++;
538     }
539
540     var activity = 0;
541     for(var x in disciplinetimetable[t][d]) {
542     activity++;
543     writeln(day, "\t",
544     (t % DayDuration)+1, "\t",
545     d, "\t",
546     x.section, "\t",
547     x.teacher, "\t",
548     x.room);
549     }
550
551     }
552     }
553     }
```


The first stage model.dat

```
1       SheetConnection sheet ("tcloud-2.xlsx");
2       TeacherDisciplineSet from SheetRead (sheet,"TeacherDisciplineSet
            !A3:C405");
3       RequirementSet from SheetRead (sheet,"RequirementSet!A4:E167");
4       MorningDiscipline from SheetRead (sheet,"etc!A4:A10");
5       AfternoonDiscipline from SheetRead (sheet,"etc!B4:B5");
6       SpecificDiscipline from SheetRead (sheet,"etc!P4:P11");
7       NeedBreak from SheetRead (sheet,"etc!D4:E8");
```

```
8       DedicatedRoomSet from SheetRead (sheet,"etc!G4:H24");
9       LabRoom from SheetRead (sheet,"etc!K4:K6");
10      SmallRoom from SheetRead (sheet,"etc!K7:K13");
11      MediumRoom from SheetRead (sheet,"etc!K14:K29");
12      LargeRoom from SheetRead (sheet,"etc!K30:K41");
13      DummyRoom from SheetRead (sheet,"etc!K42:K56");
14      RoomCapacitySet from SheetRead (sheet,"etc!K4:L48");
15      DisciplineSet from SheetRead (sheet,"DisciplineWorkload!A3:C105"
            );
16      WorkloadSet from SheetRead (sheet,"TeacherWorkload!A2:E63");
17
18      BreakDuration = 2;
19      DayDuration = 16;
20      Day = 5;
21      MeetingDay = 3;
```

APPENDIX M: The code for the second stage model

The second stage model.mod


```
1   {string} Discipline=...;
2   {string} Teacher101=...;
3   {string} Teacher103=...;
4   {string} Teacher107=...;
5   {string} Teacher113=...;
6   {string} Teacher115=...;
7   {string} Teacher117=...;
8   {string} Teacher119=...;
9   {string} Teacher675=...;
10  {string} Teacher=...;
11
12  {int} TimeSlot=...;
```

```
13
14    tuple Dsection {
15    int dtime;
16    string discipline;
17    int section;
18    }
19    tuple TDpreference {
20    string teacher;
21    string discipline;
22    int preference;
23    }
24    tuple Tworkload {
25    string teacher;
26    int requestedworkload;
27    int oldworkload;
28    }
29    tuple Dworkload {
30    string discipline;
31    int disciplineworkload;
32    }
33
34    {Dsection} DSectionSet=...;
35    {TDpreference} TDPreferenceSet=...;
36    {Tworkload} TWorkloadSet=...;
37    {Dworkload} DWorkloadSet=...;
38
39    {int} AvaiTimeA1=...;
40    {int} AvaiTimeA2=...;
41    {int} AvaiTimeA3=...;
42    {int} AvaiTimeA4=...;
43    {int} AvaiTimeA5=...;
44    {int} AvaiTimeA6=...;
45    {int} AvaiTimeA7=...;
```

```
46      {int} AvaiTimeA8=...;
47      {int} AvaiTimeA9=...;
48      {int} AvaiTimeA10=...;
49      {int} AvaiTimeA11=...;
50      {int} AvaiTimeA12=...;
51      {int} AvaiTimeA13=...;
52      {int} AvaiTimeA14=...;
53      {int} AvaiTimeA15=...;
54      {int} AvaiTimeA16=...;
55      {int} AvaiTimeA17=...;
56      {int} AvaiTimeA18=...;
57      {int} AvaiTimeA19=...;
58      {int} AvaiTimeA20=...;
59      {int} AvaiTimeA21=...;
60      {int} AvaiTimeA22=...;
61      {int} AvaiTimeA23=...;
62      {int} AvaiTimeA24=...;
63      {int} AvaiTimeA25=...;
64      {int} AvaiTimeA26=...;
65      {int} AvaiTimeA27=...;
66      {int} AvaiTimeA28=...;
67      {int} AvaiTimeA29=...;
68      {int} AvaiTimeA30=...;
69      {int} AvaiTimeA31=...;
70      {int} AvaiTimeA32=...;
71      {int} AvaiTimeA33=...;
72      {int} AvaiTimeA34=...;
73      {int} AvaiTimeA35=...;
74      {int} AvaiTimeA36=...;
75      {int} AvaiTimeA37=...;
76      {int} AvaiTimeA38=...;
77      {int} AvaiTimeA39=...;
78      {int} AvaiTimeA40=...;
```

```
79    {int} AvaiTimeA41=...;
80    {int} AvaiTimeA42=...;
81    {int} AvaiTimeA43=...;
82    {int} AvaiTimeA44=...;
83    {int} AvaiTimeA45=...;
84    {int} AvaiTimeA46=...;
85    {int} AvaiTimeA47=...;
86    {int} AvaiTimeA48=...;
87    {int} AvaiTimeA49=...;
88    {int} AvaiTimeA50=...;
89    {int} AvaiTimeA51=...;
90    {int} AvaiTimeA52=...;
91    {int} AvaiTimeA53=...;
92    {int} AvaiTimeA54=...;
93    {int} AvaiTimeA55=...;
94    {int} AvaiTimeA56=...;
95    {int} AvaiTimeA57=...;
96    {int} AvaiTimeA58=...;
97    {int} AvaiTimeA59=...;
98    {int} AvaiTimeA60=...;
99
100   {int} Section={r | <p,q,r> in DSectionSet};
101
102   {int} DisciplineTime1011=...;
103   {int} DisciplineTime1012=...;
104
105   {int} DisciplineTime1031=...;
106   {int} DisciplineTime1032=...;
107   {int} DisciplineTime1033=...;
108
109   {int} DisciplineTime1071=...;
110
111   {int} DisciplineTime1072=...;
```

```
112     {int} DisciplineTime1073=...;
113     {int} DisciplineTime1074=...;
114     {int} DisciplineTime1075=...;
115     {int} DisciplineTime1076=...;
116     {int} DisciplineTime1077=...;
117
118     {int} DisciplineTime1131=...;
119     {int} DisciplineTime1132=...;
120
121     {int} DisciplineTime1151=...;
122     {int} DisciplineTime1152=...;
123
124     {int} DisciplineTime1171=...;
125     {int} DisciplineTime1172=...;
126     {int} DisciplineTime1173=...;
127     {int} DisciplineTime1174=...;
128
129     {int} DisciplineTime1191=...;
130     {int} DisciplineTime1192=...;
131     {int} DisciplineTime1193=...;
132
133     {int} DisciplineTime6751=...;
134
135     int Preference[p in Teacher][q in Discipline] = (card({r | <p,q,r>
            in TDPreferenceSet})==0)?0:first({r | <p,q,r> in TDPreferenceSet
            });
136     int RequestedWorkload[p in Teacher] = (card({q | <p,q,r> in
            TWorkloadSet})==0)?0:first({q | <p,q,r> in TWorkloadSet});
137     int OldWorkload[p in Teacher] = (card({r | <p,q,r> in TWorkloadSet})
            ==0)?0:first({r | <p,q,r> in TWorkloadSet});
138     int DisciplineWorkload[p in Discipline] = (card({q | <p,q> in
            DWorkloadSet})==0)?0:first({q | <p,q> in DWorkloadSet});
139
```

```
140     dvar int TeachingPreference[Teacher];
141     dvar int PenaltyWorkload[Teacher];
142     dvar boolean Teaching[Teacher][Discipline];
143     dvar boolean TeachingS[Teacher][Discipline][Section];
144     dvar boolean TeachingST[Teacher][Discipline][Section][TimeSlot];
145
146     //The objective function
147     minimize sum(x in Teacher) abs(PenaltyWorkload[x]) - sum(x in
            Teacher) TeachingPreference[x];
148
149     subject to {
150     //Summation of teaching preference.
151     forall(i in Teacher)
152     TeachingPreference[i] == sum(j in Discipline) Preference[i][j]*
            Teaching[i][j];
153     //Each course section can be taught by only one instructor.
154     sum(i in Teacher101) TeachingS[i]["2301101"][1]==1;
155     sum(i in Teacher101) TeachingS[i]["2301101"][2]==1;
156
157     sum(i in Teacher103) TeachingS[i]["2301103"][1]==1;
158     sum(i in Teacher103) TeachingS[i]["2301103"][2]==1;
159     sum(i in Teacher103) TeachingS[i]["2301103"][3]==1;
160
161     sum(i in Teacher107) TeachingS[i]["2301107"][1]==1;
162     sum(i in Teacher107) TeachingS[i]["2301107"][2]==1;
163     sum(i in Teacher107) TeachingS[i]["2301107"][3]==1;
164     sum(i in Teacher107) TeachingS[i]["2301107"][4]==1;
165     sum(i in Teacher107) TeachingS[i]["2301107"][5]==1;
166     sum(i in Teacher107) TeachingS[i]["2301107"][6]==1;
167     sum(i in Teacher107) TeachingS[i]["2301107"][7]==1;
168
169     sum(i in Teacher113) TeachingS[i]["2301113"][1]==1;
170     sum(i in Teacher113) TeachingS[i]["2301113"][2]==1;
```

```
171
172     sum(i in Teacher115) TeachingS[i]["2301115"][1]==1;
173     sum(i in Teacher115) TeachingS[i]["2301115"][2]==1;
174
175     sum(i in Teacher117) TeachingS[i]["2301117"][1]==1;
176     sum(i in Teacher117) TeachingS[i]["2301117"][2]==1;
177     sum(i in Teacher117) TeachingS[i]["2301117"][3]==1;
178     sum(i in Teacher117) TeachingS[i]["2301117"][4]==1;
179
180     sum(i in Teacher119) TeachingS[i]["2301119"][1]==1;
181     sum(i in Teacher119) TeachingS[i]["2301119"][2]==1;
182     sum(i in Teacher119) TeachingS[i]["2301119"][3]==1;
183
184     sum(i in Teacher675) TeachingS[i]["2301675"][1]==1;
185
186     sum(i in Teacher) Teaching[i]["2301101"] == 2;
187     sum(i in Teacher) Teaching[i]["2301103"] == 3;
188     sum(i in Teacher) Teaching[i]["2301107"] == 7;
189     sum(i in Teacher) Teaching[i]["2301113"] == 2;
190     sum(i in Teacher) Teaching[i]["2301115"] == 2;
191     sum(i in Teacher) Teaching[i]["2301117"] == 4;
192     sum(i in Teacher) Teaching[i]["2301119"] == 3;
193     sum(i in Teacher) Teaching[i]["2301675"] == 1;
194
195     //
196     // The instructors are available at the assigned time slot. //
197     //
198
199     // Available time for discipline 2301101 sec 1
200     if(card(DisciplineTime1011 inter AvaiTimeA15)!=card(
            DisciplineTime1011))
201     TeachingS["A15"]["2301101"][1]==0;
```

```
202     if(card(DisciplineTime1011 inter AvaiTimeA17)!=card(
            DisciplineTime1011))
203     TeachingS["A17"]["2301101"][1]==0;
204     if(card(DisciplineTime1011 inter AvaiTimeA19)!=card(
            DisciplineTime1011))
205     TeachingS["A19"]["2301101"][1]==0;
206     if(card(DisciplineTime1011 inter AvaiTimeA21)!=card(
            DisciplineTime1011))
207     TeachingS["A21"]["2301101"][1]==0;
208     if(card(DisciplineTime1011 inter AvaiTimeA22)!=card(
            DisciplineTime1011))
209     TeachingS["A22"]["2301101"][1]==0;
210     if(card(DisciplineTime1011 inter AvaiTimeA23)!=card(
            DisciplineTime1011))
211     TeachingS["A23"]["2301101"][1]==0;
212     if(card(DisciplineTime1011 inter AvaiTimeA30)!=card(
            DisciplineTime1011))
213     TeachingS["A30"]["2301101"][1]==0;
214     if(card(DisciplineTime1011 inter AvaiTimeA31)!=card(
            DisciplineTime1011))
215     TeachingS["A31"]["2301101"][1]==0;
216     if(card(DisciplineTime1011 inter AvaiTimeA38)!=card(
            DisciplineTime1011))
217     TeachingS["A38"]["2301101"][1]==0;
218     if(card(DisciplineTime1011 inter AvaiTimeA39)!=card(
            DisciplineTime1011))
219     TeachingS["A39"]["2301101"][1]==0;
220     if(card(DisciplineTime1011 inter AvaiTimeA41)!=card(
            DisciplineTime1011))
221     TeachingS["A41"]["2301101"][1]==0;
222     if(card(DisciplineTime1011 inter AvaiTimeA42)!=card(
            DisciplineTime1011))
223     TeachingS["A42"]["2301101"][1]==0;
```

```
224     if(card(DisciplineTime1011 inter AvaiTimeA45)!=card(
            DisciplineTime1011))
225     TeachingS["A45"]["2301101"][1]==0;
226     if(card(DisciplineTime1011 inter AvaiTimeA46)!=card(
            DisciplineTime1011))
227     TeachingS["A46"]["2301101"][1]==0;
228     if(card(DisciplineTime1011 inter AvaiTimeA47)!=card(
            DisciplineTime1011))
229     TeachingS["A47"]["2301101"][1]==0;
230     if(card(DisciplineTime1011 inter AvaiTimeA48)!=card(
            DisciplineTime1011))
231     TeachingS["A48"]["2301101"][1]==0;
232     if(card(DisciplineTime1011 inter AvaiTimeA52)!=card(
            DisciplineTime1011))
233     TeachingS["A52"]["2301101"][1]==0;
234     //Available time for discipline 2301101 sec 2
235     if(card(DisciplineTime1012 inter AvaiTimeA15)!=card(
            DisciplineTime1012))
236     TeachingS["A15"]["2301101"][2]==0;
237     if(card(DisciplineTime1012 inter AvaiTimeA17)!=card(
            DisciplineTime1012))
238     TeachingS["A17"]["2301101"][2]==0;
239     if(card(DisciplineTime1012 inter AvaiTimeA19)!=card(
            DisciplineTime1012))
240     TeachingS["A19"]["2301101"][2]==0;
241     if(card(DisciplineTime1012 inter AvaiTimeA21)!=card(
            DisciplineTime1012))
242     TeachingS["A21"]["2301101"][2]==0;
243     if(card(DisciplineTime1011 inter AvaiTimeA22)!=card(
            DisciplineTime1012))
244     TeachingS["A22"]["2301101"][2]==0;
245     if(card(DisciplineTime1012 inter AvaiTimeA23)!=card(
            DisciplineTime1012))
```

```
246     TeachingS["A23"]["2301101"][2]==0;
247     if(card(DisciplineTime1012 inter AvaiTimeA30)!=card(
            DisciplineTime1012))
248     TeachingS["A30"]["2301101"][2]==0;
249     if(card(DisciplineTime1012 inter AvaiTimeA31)!=card(
            DisciplineTime1012))
250     TeachingS["A31"]["2301101"][2]==0;
251     if(card(DisciplineTime1012 inter AvaiTimeA38)!=card(
            DisciplineTime1012))
252     TeachingS["A38"]["2301101"][2]==0;
253     if(card(DisciplineTime1012 inter AvaiTimeA39)!=card(
            DisciplineTime1012))
254     TeachingS["A39"]["2301101"][2]==0;
255     if(card(DisciplineTime1012 inter AvaiTimeA41)!=card(
            DisciplineTime1012))
256     TeachingS["A41"]["2301101"][2]==0;
257     if(card(DisciplineTime1012 inter AvaiTimeA42)!=card(
            DisciplineTime1012))
258     TeachingS["A42"]["2301101"][2]==0;
259     if(card(DisciplineTime1012 inter AvaiTimeA45)!=card(
            DisciplineTime1012))
260     TeachingS["A45"]["2301101"][2]==0;
261     if(card(DisciplineTime1012 inter AvaiTimeA46)!=card(
            DisciplineTime1012))
262     TeachingS["A46"]["2301101"][2]==0;
263     if(card(DisciplineTime1012 inter AvaiTimeA47)!=card(
            DisciplineTime1012))
264     TeachingS["A47"]["2301101"][2]==0;
265     if(card(DisciplineTime1012 inter AvaiTimeA48)!=card(
            DisciplineTime1012))
266     TeachingS["A48"]["2301101"][2]==0;
267     if(card(DisciplineTime1012 inter AvaiTimeA52)!=card(
            DisciplineTime1012))
```

```
268     TeachingS["A52"]["2301101"][2]==0;
269     // Available time for discipline 2301103 sec 1
270     if(card(DisciplineTime1031 inter AvaiTimeA15)!=card(
            DisciplineTime1031))
271     TeachingS["A15"]["2301103"][1]==0;
272     if(card(DisciplineTime1031 inter AvaiTimeA17)!=card(
            DisciplineTime1031))
273     TeachingS["A17"]["2301103"][1]==0;
274     if(card(DisciplineTime1031 inter AvaiTimeA19)!=card(
            DisciplineTime1031))
275     TeachingS["A19"]["2301103"][1]==0;
276     if(card(DisciplineTime1031 inter AvaiTimeA21)!=card(
            DisciplineTime1031))
277     TeachingS["A21"]["2301103"][1]==0;
278     if(card(DisciplineTime1031 inter AvaiTimeA23)!=card(
            DisciplineTime1031))
279     TeachingS["A23"]["2301103"][1]==0;
280     if(card(DisciplineTime1031 inter AvaiTimeA31)!=card(
            DisciplineTime1031))
281     TeachingS["A31"]["2301103"][1]==0;
282     if(card(DisciplineTime1031 inter AvaiTimeA33)!=card(
            DisciplineTime1031))
283     TeachingS["A33"]["2301103"][1]==0;
284     if(card(DisciplineTime1031 inter AvaiTimeA39)!=card(
            DisciplineTime1031))
285     TeachingS["A39"]["2301103"][1]==0;
286     if(card(DisciplineTime1031 inter AvaiTimeA40)!=card(
            DisciplineTime1031))
287     TeachingS["A40"]["2301103"][1]==0;
288     if(card(DisciplineTime1031 inter AvaiTimeA41)!=card(
            DisciplineTime1031))
289     TeachingS["A41"]["2301103"][1]==0;
```

```
290     if(card(DisciplineTime1031 inter AvaiTimeA46)!=card(
            DisciplineTime1031))
291     TeachingS["A46"]["2301103"][1]==0;
292     if(card(DisciplineTime1031 inter AvaiTimeA47)!=card(
            DisciplineTime1031))
293     TeachingS["A47"]["2301103"][1]==0;
294     if(card(DisciplineTime1031 inter AvaiTimeA48)!=card(
            DisciplineTime1031))
295     TeachingS["A48"]["2301103"][1]==0;
296     if(card(DisciplineTime1031 inter AvaiTimeA59)!=card(
            DisciplineTime1031))
297     TeachingS["A59t"]["2301103"][1]==0;
298     // Available time for discipline 2301103 sec 2
299     if(card(DisciplineTime1032 inter AvaiTimeA15)!=card(
            DisciplineTime1032))
300     TeachingS["A15"]["2301103"][2]==0;
301     if(card(DisciplineTime1032 inter AvaiTimeA17)!=card(
            DisciplineTime1032))
302     TeachingS["A17"]["2301103"][2]==0;
303     if(card(DisciplineTime1032 inter AvaiTimeA19)!=card(
            DisciplineTime1032))
304     TeachingS["A19"]["2301103"][2]==0;
305     if(card(DisciplineTime1032 inter AvaiTimeA21)!=card(
            DisciplineTime1032))
306     TeachingS["A21"]["2301103"][2]==0;
307     if(card(DisciplineTime1032 inter AvaiTimeA23)!=card(
            DisciplineTime1032))
308     TeachingS["A23"]["2301103"][2]==0;
309     if(card(DisciplineTime1032 inter AvaiTimeA31)!=card(
            DisciplineTime1032))
310     TeachingS["A31"]["2301103"][2]==0;
311     if(card(DisciplineTime1032 inter AvaiTimeA33)!=card(
            DisciplineTime1032))
```

```
312     TeachingS["A33"]["2301103"][2]==0;
313     if(card(DisciplineTime1032 inter AvaiTimeA39)!=card(
            DisciplineTime1032))
314     TeachingS["A39"]["2301103"][2]==0;
315     if(card(DisciplineTime1032 inter AvaiTimeA41)!=card(
            DisciplineTime1032))
316     TeachingS["A41"]["2301103"][2]==0;
317     if(card(DisciplineTime1032 inter AvaiTimeA42)!=card(
            DisciplineTime1032))
318     TeachingS["A42"]["2301103"][2]==0;
319     if(card(DisciplineTime1032 inter AvaiTimeA46)!=card(
            DisciplineTime1032))
320     TeachingS["A46"]["2301103"][2]==0;
321     if(card(DisciplineTime1032 inter AvaiTimeA47)!=card(
            DisciplineTime1032))
322     TeachingS["A47"]["2301103"][2]==0;
323     if(card(DisciplineTime1032 inter AvaiTimeA48)!=card(
            DisciplineTime1032))
324     TeachingS["A48"]["2301103"][2]==0;
325     if(card(DisciplineTime1032 inter AvaiTimeA59)!=card(
            DisciplineTime1032))
326     TeachingS["A59"]["2301103"][2]==0;
327     // Available time for discipline 2301103 sec 3
328     if(card(DisciplineTime1033 inter AvaiTimeA15)!=card(
            DisciplineTime1033))
329     TeachingS["A15"]["2301103"][3]==0;
330     if(card(DisciplineTime1033 inter AvaiTimeA17)!=card(
            DisciplineTime1033))
331     TeachingS["A17"]["2301103"][3]==0;
332     if(card(DisciplineTime1033 inter AvaiTimeA19)!=card(
            DisciplineTime1033))
333     TeachingS["A19"]["2301103"][3]==0;
```

```
334     if(card(DisciplineTime1033 inter AvaiTimeA21)!=card(
            DisciplineTime1033))
335     TeachingS["A21"]["2301103"][3]==0;
336     if(card(DisciplineTime1033 inter AvaiTimeA23)!=card(
            DisciplineTime1033))
337     TeachingS["A23"]["2301103"][3]==0;
338     if(card(DisciplineTime1033 inter AvaiTimeA31)!=card(
            DisciplineTime1033))
339     TeachingS["A31"]["2301103"][3]==0;
340     if(card(DisciplineTime1033 inter AvaiTimeA33)!=card(
            DisciplineTime1033))
341     TeachingS["A33"]["2301103"][3]==0;
342     if(card(DisciplineTime1033 inter AvaiTimeA39)!=card(
            DisciplineTime1033))
343     TeachingS["A39"]["2301103"][3]==0;
344     if(card(DisciplineTime1033 inter AvaiTimeA41)!=card(
            DisciplineTime1033))
345     TeachingS["A41"]["2301103"][3]==0;
346     if(card(DisciplineTime1033 inter AvaiTimeA42)!=card(
            DisciplineTime1033))
347     TeachingS["A42"]["2301103"][3]==0;
348     if(card(DisciplineTime1033 inter AvaiTimeA46)!=card(
            DisciplineTime1033))
349     TeachingS["A46"]["2301103"][3]==0;
350     if(card(DisciplineTime1033 inter AvaiTimeA47)!=card(
            DisciplineTime1033))
351     TeachingS["A47"]["2301103"][3]==0;
352     if(card(DisciplineTime1033 inter AvaiTimeA48)!=card(
            DisciplineTime1033))
353     TeachingS["A48"]["2301103"][3]==0;
354     if(card(DisciplineTime1033 inter AvaiTimeA59)!=card(
            DisciplineTime1033))
355     TeachingS["A59"]["2301103"][3]==0;
```

```
356     // Available time for discipline 2301107 sec 1
357     if(card(DisciplineTime1071 inter AvaiTimeA2)!=card(
            DisciplineTime1071))
358     TeachingS["A2"]["2301107"][1]==0;
359     if(card(DisciplineTime1071 inter AvaiTimeA5)!=card(
            DisciplineTime1071))
360     TeachingS["A5"]["2301107"][1]==0;
361     if(card(DisciplineTime1071 inter AvaiTimeA6)!=card(
            DisciplineTime1071))
362     TeachingS["A6"]["2301107"][1]==0;
363     if(card(DisciplineTime1071 inter AvaiTimeA8)!=card(
            DisciplineTime1071))
364     TeachingS["A8"]["2301107"][1]==0;
365     if(card(DisciplineTime1071 inter AvaiTimeA13)!=card(
            DisciplineTime1071))
366     TeachingS["A13"]["2301107"][1]==0;
367     if(card(DisciplineTime1071 inter AvaiTimeA15)!=card(
            DisciplineTime1071))
368     TeachingS["A15"]["2301107"][1]==0;
369     if(card(DisciplineTime1071 inter AvaiTimeA18)!=card(
            DisciplineTime1071))
370     TeachingS["A18"]["2301107"][1]==0;
371     if(card(DisciplineTime1071 inter AvaiTimeA19)!=card(
            DisciplineTime1071))
372     TeachingS["A19"]["2301107"][1]==0;
373     if(card(DisciplineTime1071 inter AvaiTimeA20)!=card(
            DisciplineTime1071))
374     TeachingS["A20"]["2301107"][1]==0;
375     if(card(DisciplineTime1071 inter AvaiTimeA21)!=card(
            DisciplineTime1071))
376     TeachingS["A21"]["2301107"][1]==0;
377     if(card(DisciplineTime1071 inter AvaiTimeA25)!=card(
            DisciplineTime1071))
```

```
378     TeachingS["A25"]["2301107"][1]==0;
379     if(card(DisciplineTime1071 inter AvaiTimeA28)!=card(
            DisciplineTime1071))
380     TeachingS["A28"]["2301107"][1]==0;
381     if(card(DisciplineTime1071 inter AvaiTimeA30)!=card(
            DisciplineTime1071))
382     TeachingS["A30"]["2301107"][1]==0;
383     if(card(DisciplineTime1071 inter AvaiTimeA32)!=card(
            DisciplineTime1071))
384     TeachingS["A32"]["2301107"][1]==0;
385     if(card(DisciplineTime1071 inter AvaiTimeA33)!=card(
            DisciplineTime1071))
386     TeachingS["A33"]["2301107"][1]==0;
387     if(card(DisciplineTime1071 inter AvaiTimeA34)!=card(
            DisciplineTime1071))
388     TeachingS["A34"]["2301107"][1]==0;
389     if(card(DisciplineTime1071 inter AvaiTimeA38)!=card(
            DisciplineTime1071))
390     TeachingS["A38"]["2301107"][1]==0;
391     if(card(DisciplineTime1071 inter AvaiTimeA39)!=card(
            DisciplineTime1071))
392     TeachingS["A39"]["2301107"][1]==0;
393     if(card(DisciplineTime1071 inter AvaiTimeA41)!=card(
            DisciplineTime1071))
394     TeachingS["A41"]["2301107"][1]==0;
395     if(card(DisciplineTime1071 inter AvaiTimeA42)!=card(
            DisciplineTime1071))
396     TeachingS["A42"]["2301107"][1]==0;
397     if(card(DisciplineTime1071 inter AvaiTimeA44)!=card(
            DisciplineTime1071))
398     TeachingS["A44"]["2301107"][1]==0;
399     if(card(DisciplineTime1071 inter AvaiTimeA45)!=card(
            DisciplineTime1071))
```

```
400     TeachingS["A45"]["2301107"][1]==0;
401     if(card(DisciplineTime1071 inter AvaiTimeA47)!=card(
            DisciplineTime1071))
402     TeachingS["A47"]["2301107"][1]==0;
403     if(card(DisciplineTime1071 inter AvaiTimeA54)!=card(
            DisciplineTime1071))
404     TeachingS["A54"]["2301107"][1]==0;
405     if(card(DisciplineTime1071 inter AvaiTimeA55)!=card(
            DisciplineTime1071))
406     TeachingS["A55"]["2301107"][1]==0;
407     if(card(DisciplineTime1071 inter AvaiTimeA59)!=card(
            DisciplineTime1071))
408     TeachingS["A59"]["2301107"][1]==0;
409     if(card(DisciplineTime1071 inter AvaiTimeA43)!=card(
            DisciplineTime1071))
410     TeachingS["A43"]["2301107"][1]==0;
411     // Available time for discipline 2301107 sec 2
412     if(card(DisciplineTime1072 inter AvaiTimeA2)!=card(
            DisciplineTime1072))
413     TeachingS["A2"]["2301107"][2]==0;
414     if(card(DisciplineTime1072 inter AvaiTimeA5)!=card(
            DisciplineTime1072))
415     TeachingS["A5"]["2301107"][2]==0;
416     if(card(DisciplineTime1072 inter AvaiTimeA6)!=card(
            DisciplineTime1072))
417     TeachingS["A6"]["2301107"][2]==0;
418     if(card(DisciplineTime1072 inter AvaiTimeA8)!=card(
            DisciplineTime1072))
419     TeachingS["A8"]["2301107"][2]==0;
420     if(card(DisciplineTime1072 inter AvaiTimeA13)!=card(
            DisciplineTime1072))
421     TeachingS["A13"]["2301107"][2]==0;
```

```
422     if(card(DisciplineTime1072 inter AvaiTimeA15)!=card(
            DisciplineTime1072))
423     TeachingS["A15"]["2301107"][2]==0;
424     if(card(DisciplineTime1072 inter AvaiTimeA18)!=card(
            DisciplineTime1072))
425     TeachingS["A18"]["2301107"][2]==0;
426     if(card(DisciplineTime1072 inter AvaiTimeA19)!=card(
            DisciplineTime1072))
427     TeachingS["A19"]["2301107"][2]==0;
428     if(card(DisciplineTime1072 inter AvaiTimeA20)!=card(
            DisciplineTime1072))
429     TeachingS["A20"]["2301107"][2]==0;
430     if(card(DisciplineTime1072 inter AvaiTimeA21)!=card(
            DisciplineTime1072))
431     TeachingS["A21"]["2301107"][2]==0;
432     if(card(DisciplineTime1072 inter AvaiTimeA28)!=card(
            DisciplineTime1072))
433     TeachingS["A28"]["2301107"][2]==0;
434     if(card(DisciplineTime1072 inter AvaiTimeA30)!=card(
            DisciplineTime1072))
435     TeachingS["A30"]["2301107"][2]==0;
436     if(card(DisciplineTime1072 inter AvaiTimeA32)!=card(
            DisciplineTime1072))
437     TeachingS["A32"]["2301107"][2]==0;
438     if(card(DisciplineTime1072 inter AvaiTimeA33)!=card(
            DisciplineTime1072))
439     TeachingS["A33"]["2301107"][2]==0;
440     if(card(DisciplineTime1072 inter AvaiTimeA34)!=card(
            DisciplineTime1072))
441     TeachingS["A34"]["2301107"][2]==0;
442     if(card(DisciplineTime1072 inter AvaiTimeA38)!=card(
            DisciplineTime1072))
443     TeachingS["A38"]["2301107"][2]==0;
```

```
444     if(card(DisciplineTime1072 inter AvaiTimeA39)!=card(
            DisciplineTime1072))
445     TeachingS["A39"]["2301107"][2]==0;
446     if(card(DisciplineTime1072 inter AvaiTimeA41)!=card(
            DisciplineTime1072))
447     TeachingS["A41"]["2301107"][2]==0;
448     if(card(DisciplineTime1072 inter AvaiTimeA42)!=card(
            DisciplineTime1072))
449     TeachingS["A42"]["2301107"][2]==0;
450     if(card(DisciplineTime1072 inter AvaiTimeA44)!=card(
            DisciplineTime1072))
451     TeachingS["A44"]["2301107"][2]==0;
452     if(card(DisciplineTime1072 inter AvaiTimeA45)!=card(
            DisciplineTime1072))
453     TeachingS["A45"]["2301107"][2]==0;
454     if(card(DisciplineTime1072 inter AvaiTimeA47)!=card(
            DisciplineTime1072))
455     TeachingS["A47"]["2301107"][2]==0;
456     if(card(DisciplineTime1072 inter AvaiTimeA54)!=card(
            DisciplineTime1072))
457     TeachingS["A54"]["2301107"][2]==0;
458     if(card(DisciplineTime1072 inter AvaiTimeA55)!=card(
            DisciplineTime1072))
459     TeachingS["A55"]["2301107"][2]==0;
460     if(card(DisciplineTime1072 inter AvaiTimeA59)!=card(
            DisciplineTime1072))
461     TeachingS["A59"]["2301107"][2]==0;
462     if(card(DisciplineTime1072 inter AvaiTimeA25)!=card(
            DisciplineTime1072))
463     TeachingS["A25"]["2301107"][2]==0;
464     if(card(DisciplineTime1072 inter AvaiTimeA43)!=card(
            DisciplineTime1072))
465     TeachingS["A43"]["2301107"][2]==0;
```

```
466     // Available time for discipline 2301107 sec 3
467     if(card(DisciplineTime1073 inter AvaiTimeA2)!=card(
            DisciplineTime1073))
468     TeachingS["A2"]["2301107"][3]==0;
469     if(card(DisciplineTime1073 inter AvaiTimeA5)!=card(
            DisciplineTime1073))
470     TeachingS["A5"]["2301107"][3]==0;
471     if(card(DisciplineTime1073 inter AvaiTimeA6)!=card(
            DisciplineTime1073))
472     TeachingS["A6"]["2301107"][3]==0;
473     if(card(DisciplineTime1073 inter AvaiTimeA8)!=card(
            DisciplineTime1073))
474     TeachingS["A8"]["2301107"][3]==0;
475     if(card(DisciplineTime1073 inter AvaiTimeA13)!=card(
            DisciplineTime1073))
476     TeachingS["A13"]["2301107"][3]==0;
477     if(card(DisciplineTime1073 inter AvaiTimeA15)!=card(
            DisciplineTime1073))
478     TeachingS["A15"]["2301107"][3]==0;
479     if(card(DisciplineTime1073 inter AvaiTimeA18)!=card(
            DisciplineTime1073))
480     TeachingS["A18"]["2301107"][3]==0;
481     if(card(DisciplineTime1073 inter AvaiTimeA19)!=card(
            DisciplineTime1073))
482     TeachingS["A19"]["2301107"][3]==0;
483     if(card(DisciplineTime1073 inter AvaiTimeA20)!=card(
            DisciplineTime1073))
484     TeachingS["A20"]["2301107"][3]==0;
485     if(card(DisciplineTime1073 inter AvaiTimeA21)!=card(
            DisciplineTime1073))
486     TeachingS["A21"]["2301107"][3]==0;
487     if(card(DisciplineTime1073 inter AvaiTimeA28)!=card(
            DisciplineTime1073))
```

```
488     TeachingS["A28"]["2301107"][3]==0;
489     if(card(DisciplineTime1073 inter AvaiTimeA30)!=card(
            DisciplineTime1073))
490     TeachingS["A30"]["2301107"][3]==0;
491     if(card(DisciplineTime1073 inter AvaiTimeA32)!=card(
            DisciplineTime1073))
492     TeachingS["A32"]["2301107"][3]==0;
493     if(card(DisciplineTime1073 inter AvaiTimeA33)!=card(
            DisciplineTime1073))
494     TeachingS["A33"]["2301107"][3]==0;
495     if(card(DisciplineTime1073 inter AvaiTimeA34)!=card(
            DisciplineTime1073))
496     TeachingS["A34"]["2301107"][3]==0;
497     if(card(DisciplineTime1073 inter AvaiTimeA38)!=card(
            DisciplineTime1073))
498     TeachingS["A38"]["2301107"][3]==0;
499     if(card(DisciplineTime1073 inter AvaiTimeA39)!=card(
            DisciplineTime1073))
500     TeachingS["A39"]["2301107"][3]==0;
501     if(card(DisciplineTime1073 inter AvaiTimeA41)!=card(
            DisciplineTime1073))
502     TeachingS["A41"]["2301107"][3]==0;
503     if(card(DisciplineTime1073 inter AvaiTimeA42)!=card(
            DisciplineTime1073))
504     TeachingS["A42"]["2301107"][3]==0;
505     if(card(DisciplineTime1073 inter AvaiTimeA44)!=card(
            DisciplineTime1073))
506     TeachingS["A44"]["2301107"][3]==0;
507     if(card(DisciplineTime1073 inter AvaiTimeA45)!=card(
            DisciplineTime1073))
508     TeachingS["A45"]["2301107"][3]==0;
509     if(card(DisciplineTime1073 inter AvaiTimeA47)!=card(
            DisciplineTime1073))
```

```
510     TeachingS["A47"]["2301107"][3]==0;
511     if(card(DisciplineTime1073 inter AvaiTimeA54)!=card(
            DisciplineTime1073))
512     TeachingS["A54"]["2301107"][3]==0;
513     if(card(DisciplineTime1073 inter AvaiTimeA55)!=card(
            DisciplineTime1073))
514     TeachingS["A55"]["2301107"][3]==0;
515     if(card(DisciplineTime1073 inter AvaiTimeA59)!=card(
            DisciplineTime1073))
516     TeachingS["A59"]["2301107"][3]==0;
517     if(card(DisciplineTime1073 inter AvaiTimeA25)!=card(
            DisciplineTime1073))
518     TeachingS["A25"]["2301107"][3]==0;
519     if(card(DisciplineTime1073 inter AvaiTimeA43)!=card(
            DisciplineTime1073))
520     TeachingS["A43"]["2301107"][3]==0;
521     // Available time for discipline 2301107 sec 4
522     if(card(DisciplineTime1074 inter AvaiTimeA2)!=card(
            DisciplineTime1074))
523     TeachingS["A2"]["2301107"][4]==0;
524     if(card(DisciplineTime1074 inter AvaiTimeA5)!=card(
            DisciplineTime1074))
525     TeachingS["A5"]["2301107"][4]==0;
526     if(card(DisciplineTime1074 inter AvaiTimeA6)!=card(
            DisciplineTime1074))
527     TeachingS["A6"]["2301107"][4]==0;
528     if(card(DisciplineTime1074 inter AvaiTimeA8)!=card(
            DisciplineTime1074))
529     TeachingS["A8"]["2301107"][4]==0;
530     if(card(DisciplineTime1074 inter AvaiTimeA13)!=card(
            DisciplineTime1074))
531     TeachingS["A13"]["2301107"][4]==0;
```

```
532     if(card(DisciplineTime1074 inter AvaiTimeA15)!=card(
            DisciplineTime1074))
533     TeachingS["A15"]["2301107"][4]==0;
534     if(card(DisciplineTime1074 inter AvaiTimeA18)!=card(
            DisciplineTime1074))
535     TeachingS["A18"]["2301107"][4]==0;
536     if(card(DisciplineTime1074 inter AvaiTimeA19)!=card(
            DisciplineTime1074))
537     TeachingS["A19"]["2301107"][4]==0;
538     if(card(DisciplineTime1074 inter AvaiTimeA20)!=card(
            DisciplineTime1074))
539     TeachingS["A20"]["2301107"][4]==0;
540     if(card(DisciplineTime1074 inter AvaiTimeA21)!=card(
            DisciplineTime1074))
541     TeachingS["A21"]["2301107"][4]==0;
542     if(card(DisciplineTime1074 inter AvaiTimeA28)!=card(
            DisciplineTime1074))
543     TeachingS["A28"]["2301107"][4]==0;
544     if(card(DisciplineTime1074 inter AvaiTimeA30)!=card(
            DisciplineTime1074))
545     TeachingS["A30"]["2301107"][4]==0;
546     if(card(DisciplineTime1074 inter AvaiTimeA32)!=card(
            DisciplineTime1074))
547     TeachingS["A32"]["2301107"][4]==0;
548     if(card(DisciplineTime1074 inter AvaiTimeA33)!=card(
            DisciplineTime1074))
549     TeachingS["A33"]["2301107"][4]==0;
550     if(card(DisciplineTime1074 inter AvaiTimeA34)!=card(
            DisciplineTime1074))
551     TeachingS["A34"]["2301107"][4]==0;
552     if(card(DisciplineTime1074 inter AvaiTimeA38)!=card(
            DisciplineTime1074))
553     TeachingS["A38"]["2301107"][4]==0;
```

```
554     if(card(DisciplineTime1074 inter AvaiTimeA39)!=card(
            DisciplineTime1074))
555     TeachingS["A39"]["2301107"][4]==0;
556     if(card(DisciplineTime1074 inter AvaiTimeA41)!=card(
            DisciplineTime1074))
557     TeachingS["A41"]["2301107"][4]==0;
558     if(card(DisciplineTime1074 inter AvaiTimeA42)!=card(
            DisciplineTime1074))
559     TeachingS["A42"]["2301107"][4]==0;
560     if(card(DisciplineTime1074 inter AvaiTimeA44)!=card(
            DisciplineTime1074))
561     TeachingS["A44"]["2301107"][4]==0;
562     if(card(DisciplineTime1074 inter AvaiTimeA45)!=card(
            DisciplineTime1074))
563     TeachingS["A45"]["2301107"][4]==0;
564     if(card(DisciplineTime1074 inter AvaiTimeA47)!=card(
            DisciplineTime1074))
565     TeachingS["A47"]["2301107"][4]==0;
566     if(card(DisciplineTime1074 inter AvaiTimeA54)!=card(
            DisciplineTime1074))
567     TeachingS["A54"]["2301107"][4]==0;
568     if(card(DisciplineTime1074 inter AvaiTimeA55)!=card(
            DisciplineTime1074))
569     TeachingS["A55"]["2301107"][4]==0;
570     if(card(DisciplineTime1074 inter AvaiTimeA59)!=card(
            DisciplineTime1074))
571     TeachingS["A59"]["2301107"][4]==0;
572     if(card(DisciplineTime1074 inter AvaiTimeA25)!=card(
            DisciplineTime1074))
573     TeachingS["A25"]["2301107"][4]==0;
574     if(card(DisciplineTime1074 inter AvaiTimeA43)!=card(
            DisciplineTime1074))
575     TeachingS["A43"]["2301107"][4]==0;
```

```
576     // Available time for discipline 2301107 sec 5
577     if(card(DisciplineTime1075 inter AvaiTimeA2)!=card(
            DisciplineTime1075))
578     TeachingS["A2"]["2301107"][5]==0;
579     if(card(DisciplineTime1075 inter AvaiTimeA5)!=card(
            DisciplineTime1075))
580     TeachingS["A5"]["2301107"][5]==0;
581     if(card(DisciplineTime1075 inter AvaiTimeA6)!=card(
            DisciplineTime1075))
582     TeachingS["A6"]["2301107"][5]==0;
583     if(card(DisciplineTime1075 inter AvaiTimeA8)!=card(
            DisciplineTime1075))
584     TeachingS["A8"]["2301107"][5]==0;
585     if(card(DisciplineTime1075 inter AvaiTimeA13)!=card(
            DisciplineTime1075))
586     TeachingS["A13"]["2301107"][5]==0;
587     if(card(DisciplineTime1075 inter AvaiTimeA15)!=card(
            DisciplineTime1075))
588     TeachingS["A15"]["2301107"][5]==0;
589     if(card(DisciplineTime1075 inter AvaiTimeA18)!=card(
            DisciplineTime1075))
590     TeachingS["A18"]["2301107"][5]==0;
591     if(card(DisciplineTime1075 inter AvaiTimeA19)!=card(
            DisciplineTime1075))
592     TeachingS["A19"]["2301107"][5]==0;
593     if(card(DisciplineTime1075 inter AvaiTimeA20)!=card(
            DisciplineTime1075))
594     TeachingS["A20"]["2301107"][5]==0;
595     if(card(DisciplineTime1075 inter AvaiTimeA21)!=card(
            DisciplineTime1075))
596     TeachingS["A21"]["2301107"][5]==0;
597     if(card(DisciplineTime1075 inter AvaiTimeA28)!=card(
            DisciplineTime1075))
```

```
598     TeachingS["A28"]["2301107"][5]==0;
599     if(card(DisciplineTime1075 inter AvaiTimeA30)!=card(
            DisciplineTime1075))
600     TeachingS["A30"]["2301107"][5]==0;
601     if(card(DisciplineTime1075 inter AvaiTimeA32)!=card(
            DisciplineTime1075))
602     TeachingS["A32"]["2301107"][5]==0;
603     if(card(DisciplineTime1075 inter AvaiTimeA33)!=card(
            DisciplineTime1075))
604     TeachingS["A33"]["2301107"][5]==0;
605     if(card(DisciplineTime1075 inter AvaiTimeA34)!=card(
            DisciplineTime1075))
606     TeachingS["A34"]["2301107"][5]==0;
607     if(card(DisciplineTime1075 inter AvaiTimeA38)!=card(
            DisciplineTime1075))
608     TeachingS["A38"]["2301107"][5]==0;
609     if(card(DisciplineTime1075 inter AvaiTimeA39)!=card(
            DisciplineTime1075))
610     TeachingS["A39"]["2301107"][5]==0;
611     if(card(DisciplineTime1075 inter AvaiTimeA41)!=card(
            DisciplineTime1075))
612     TeachingS["A41"]["2301107"][5]==0;
613     if(card(DisciplineTime1075 inter AvaiTimeA42)!=card(
            DisciplineTime1075))
614     TeachingS["A42"]["2301107"][5]==0;
615     if(card(DisciplineTime1075 inter AvaiTimeA44)!=card(
            DisciplineTime1075))
616     TeachingS["A44"]["2301107"][5]==0;
617     if(card(DisciplineTime1075 inter AvaiTimeA45)!=card(
            DisciplineTime1075))
618     TeachingS["A45"]["2301107"][5]==0;
619     if(card(DisciplineTime1075 inter AvaiTimeA47)!=card(
            DisciplineTime1075))
```

```
620     TeachingS["A47"]["2301107"][5]==0;
621     if(card(DisciplineTime1075 inter AvaiTimeA54)!=card(
            DisciplineTime1075))
622     TeachingS["A54"]["2301107"][5]==0;
623     if(card(DisciplineTime1075 inter AvaiTimeA55)!=card(
            DisciplineTime1075))
624     TeachingS["A55"]["2301107"][5]==0;
625     if(card(DisciplineTime1075 inter AvaiTimeA59)!=card(
            DisciplineTime1075))
626     TeachingS["A59"]["2301107"][5]==0;
627     if(card(DisciplineTime1075 inter AvaiTimeA25)!=card(
            DisciplineTime1075))
628     TeachingS["A25"]["2301107"][5]==0;
629     if(card(DisciplineTime1075 inter AvaiTimeA43)!=card(
            DisciplineTime1075))
630     TeachingS["A43"]["2301107"][5]==0;
631     // Available time for discipline 2301107 sec 6
632     if(card(DisciplineTime1076 inter AvaiTimeA2)!=card(
            DisciplineTime1076))
633     TeachingS["A2"]["2301107"][6]==0;
634     if(card(DisciplineTime1076 inter AvaiTimeA5)!=card(
            DisciplineTime1076))
635     TeachingS["A5"]["2301107"][6]==0;
636     if(card(DisciplineTime1076 inter AvaiTimeA6)!=card(
            DisciplineTime1076))
637     TeachingS["A6"]["2301107"][6]==0;
638     if(card(DisciplineTime1076 inter AvaiTimeA8)!=card(
            DisciplineTime1076))
639     TeachingS["A8"]["2301107"][6]==0;
640     if(card(DisciplineTime1076 inter AvaiTimeA13)!=card(
            DisciplineTime1076))
641     TeachingS["A13"]["2301107"][6]==0;
```

```
642     if(card(DisciplineTime1076 inter AvaiTimeA15)!=card(
            DisciplineTime1076))
643     TeachingS["A15"]["2301107"][6]==0;
644     if(card(DisciplineTime1076 inter AvaiTimeA18)!=card(
            DisciplineTime1076))
645     TeachingS["A18"]["2301107"][6]==0;
646     if(card(DisciplineTime1076 inter AvaiTimeA19)!=card(
            DisciplineTime1076))
647     TeachingS["A19"]["2301107"][6]==0;
648     if(card(DisciplineTime1076 inter AvaiTimeA20)!=card(
            DisciplineTime1076))
649     TeachingS["A20"]["2301107"][6]==0;
650     if(card(DisciplineTime1076 inter AvaiTimeA21)!=card(
            DisciplineTime1076))
651     TeachingS["A21"]["2301107"][6]==0;
652     if(card(DisciplineTime1076 inter AvaiTimeA28)!=card(
            DisciplineTime1076))
653     TeachingS["A28"]["2301107"][6]==0;
654     if(card(DisciplineTime1076 inter AvaiTimeA30)!=card(
            DisciplineTime1076))
655     TeachingS["A30"]["2301107"][6]==0;
656     if(card(DisciplineTime1076 inter AvaiTimeA32)!=card(
            DisciplineTime1076))
657     TeachingS["A32"]["2301107"][6]==0;
658     if(card(DisciplineTime1076 inter AvaiTimeA33)!=card(
            DisciplineTime1076))
659     TeachingS["A33"]["2301107"][6]==0;
660     if(card(DisciplineTime1076 inter AvaiTimeA34)!=card(
            DisciplineTime1076))
661     TeachingS["A34"]["2301107"][6]==0;
662     if(card(DisciplineTime1076 inter AvaiTimeA38)!=card(
            DisciplineTime1076))
663     TeachingS["A38"]["2301107"][6]==0;
```

```
664     if(card(DisciplineTime1076 inter AvaiTimeA39)!=card(
            DisciplineTime1076))
665     TeachingS["A39"]["2301107"][6]==0;
666     if(card(DisciplineTime1076 inter AvaiTimeA41)!=card(
            DisciplineTime1076))
667     TeachingS["A41"]["2301107"][6]==0;
668     if(card(DisciplineTime1076 inter AvaiTimeA42)!=card(
            DisciplineTime1076))
669     TeachingS["A42"]["2301107"][6]==0;
670     if(card(DisciplineTime1076 inter AvaiTimeA44)!=card(
            DisciplineTime1076))
671     TeachingS["A44"]["2301107"][6]==0;
672     if(card(DisciplineTime1076 inter AvaiTimeA45)!=card(
            DisciplineTime1076))
673     TeachingS["A45"]["2301107"][6]==0;
674     if(card(DisciplineTime1076 inter AvaiTimeA47)!=card(
            DisciplineTime1076))
675     TeachingS["A47"]["2301107"][6]==0;
676     if(card(DisciplineTime1076 inter AvaiTimeA54)!=card(
            DisciplineTime1076))
677     TeachingS["A54"]["2301107"][6]==0;
678     if(card(DisciplineTime1076 inter AvaiTimeA55)!=card(
            DisciplineTime1076))
679     TeachingS["A55"]["2301107"][6]==0;
680     if(card(DisciplineTime1076 inter AvaiTimeA59)!=card(
            DisciplineTime1076))
681     TeachingS["A59"]["2301107"][6]==0;
682     if(card(DisciplineTime1076 inter AvaiTimeA25)!=card(
            DisciplineTime1076))
683     TeachingS["A25"]["2301107"][6]==0;
684     if(card(DisciplineTime1076 inter AvaiTimeA43)!=card(
            DisciplineTime1076))
685     TeachingS["A43"]["2301107"][6]==0;
```

```
686     // Available time for discipline 2301107 sec 2
687     if(card(DisciplineTime1077 inter AvaiTimeA2)!=card(
            DisciplineTime1077))
688     TeachingS["A2"]["2301107"][7]==0;
689     if(card(DisciplineTime1077 inter AvaiTimeA5)!=card(
            DisciplineTime1077))
690     TeachingS["A5"]["2301107"][7]==0;
691     if(card(DisciplineTime1077 inter AvaiTimeA6)!=card(
            DisciplineTime1077))
692     TeachingS["A6"]["2301107"][7]==0;
693     if(card(DisciplineTime1077 inter AvaiTimeA8)!=card(
            DisciplineTime1077))
694     TeachingS["A8"]["2301107"][7]==0;
695     if(card(DisciplineTime1077 inter AvaiTimeA13)!=card(
            DisciplineTime1077))
696     TeachingS["A13"]["2301107"][7]==0;
697     if(card(DisciplineTime1077 inter AvaiTimeA15)!=card(
            DisciplineTime1077))
698     TeachingS["A15"]["2301107"][7]==0;
699     if(card(DisciplineTime1077 inter AvaiTimeA18)!=card(
            DisciplineTime1077))
700     TeachingS["A18"]["2301107"][7]==0;
701     if(card(DisciplineTime1077 inter AvaiTimeA19)!=card(
            DisciplineTime1077))
702     TeachingS["A19"]["2301107"][7]==0;
703     if(card(DisciplineTime1077 inter AvaiTimeA20)!=card(
            DisciplineTime1077))
704     TeachingS["A20"]["2301107"][7]==0;
705     if(card(DisciplineTime1077 inter AvaiTimeA21)!=card(
            DisciplineTime1077))
706     TeachingS["A21"]["2301107"][7]==0;
707     if(card(DisciplineTime1077 inter AvaiTimeA28)!=card(
            DisciplineTime1077))
```

```
708     TeachingS["A28"]["2301107"][7]==0;
709     if(card(DisciplineTime1077 inter AvaiTimeA30)!=card(
            DisciplineTime1077))
710     TeachingS["A30"]["2301107"][7]==0;
711     if(card(DisciplineTime1077 inter AvaiTimeA32)!=card(
            DisciplineTime1077))
712     TeachingS["A32"]["2301107"][7]==0;
713     if(card(DisciplineTime1077 inter AvaiTimeA33)!=card(
            DisciplineTime1077))
714     TeachingS["A33"]["2301107"][7]==0;
715     if(card(DisciplineTime1077 inter AvaiTimeA34)!=card(
            DisciplineTime1077))
716     TeachingS["A34"]["2301107"][7]==0;
717     if(card(DisciplineTime1077 inter AvaiTimeA38)!=card(
            DisciplineTime1077))
718     TeachingS["A38"]["2301107"][7]==0;
719     if(card(DisciplineTime1077 inter AvaiTimeA39)!=card(
            DisciplineTime1077))
720     TeachingS["A39"]["2301107"][7]==0;
721     if(card(DisciplineTime1077 inter AvaiTimeA41)!=card(
            DisciplineTime1077))
722     TeachingS["A41"]["2301107"][7]==0;
723     if(card(DisciplineTime1077 inter AvaiTimeA42)!=card(
            DisciplineTime1077))
724     TeachingS["A42"]["2301107"][7]==0;
725     if(card(DisciplineTime1077 inter AvaiTimeA44)!=card(
            DisciplineTime1077))
726     TeachingS["A44"]["2301107"][7]==0;
727     if(card(DisciplineTime1077 inter AvaiTimeA45)!=card(
            DisciplineTime1077))
728     TeachingS["A45"]["2301107"][7]==0;
729     if(card(DisciplineTime1077 inter AvaiTimeA47)!=card(
            DisciplineTime1077))
```

```
730     TeachingS["A47"]["2301107"][7]==0;
731     if(card(DisciplineTime1077 inter AvaiTimeA54)!=card(
            DisciplineTime1077))
732     TeachingS["A54"]["2301107"][7]==0;
733     if(card(DisciplineTime1077 inter AvaiTimeA55)!=card(
            DisciplineTime1077))
734     TeachingS["A55"]["2301107"][7]==0;
735     if(card(DisciplineTime1077 inter AvaiTimeA59)!=card(
            DisciplineTime1077))
736     TeachingS["A59"]["2301107"][7]==0;
737     if(card(DisciplineTime1077 inter AvaiTimeA25)!=card(
            DisciplineTime1077))
738     TeachingS["A25"]["2301107"][7]==0;
739     if(card(DisciplineTime1077 inter AvaiTimeA43)!=card(
            DisciplineTime1077))
740     TeachingS["A43"]["2301107"][7]==0;
741     // Available time for discipline 2301113 sec 1
742     if(card(DisciplineTime1131 inter AvaiTimeA6)!=card(
            DisciplineTime1131))
743     TeachingS["A6"]["2301113"][1]==0;
744     if(card(DisciplineTime1131 inter AvaiTimeA19)!=card(
            DisciplineTime1131))
745     TeachingS["A19"]["2301113"][1]==0;
746     if(card(DisciplineTime1131 inter AvaiTimeA20)!=card(
            DisciplineTime1131))
747     TeachingS["A20"]["2301113"][1]==0;
748     if(card(DisciplineTime1131 inter AvaiTimeA21)!=card(
            DisciplineTime1131))
749     TeachingS["A21"]["2301113"][1]==0;
750     if(card(DisciplineTime1131 inter AvaiTimeA22)!=card(
            DisciplineTime1131))
751     TeachingS["A22"]["2301113"][1]==0;
```

```
752     if(card(DisciplineTime1131 inter AvaiTimeA23)!=card(
            DisciplineTime1131))
753     TeachingS["A23"]["2301113"][1]==0;
754     if(card(DisciplineTime1131 inter AvaiTimeA31)!=card(
            DisciplineTime1131))
755     TeachingS["A31"]["2301113"][1]==0;
756     if(card(DisciplineTime1131 inter AvaiTimeA41)!=card(
            DisciplineTime1131))
757     TeachingS["A41"]["2301113"][1]==0;
758     if(card(DisciplineTime1131 inter AvaiTimeA42)!=card(
            DisciplineTime1131))
759     TeachingS["A42"]["2301113"][1]==0;
760     if(card(DisciplineTime1131 inter AvaiTimeA46)!=card(
            DisciplineTime1131))
761     TeachingS["A46"]["2301113"][1]==0;
762     if(card(DisciplineTime1131 inter AvaiTimeA48)!=card(
            DisciplineTime1131))
763     TeachingS["A48"]["2301113"][1]==0;
764     if(card(DisciplineTime1131 inter AvaiTimeA57)!=card(
            DisciplineTime1131))
765     TeachingS["A57"]["2301113"][1]==0;
766     // Available time for discipline 2301113 sec 2
767     if(card(DisciplineTime1132 inter AvaiTimeA6)!=card(
            DisciplineTime1132))
768     TeachingS["A6"]["2301113"][2]==0;
769     if(card(DisciplineTime1132 inter AvaiTimeA19)!=card(
            DisciplineTime1132))
770     TeachingS["A19"]["2301113"][2]==0;
771     if(card(DisciplineTime1132 inter AvaiTimeA20)!=card(
            DisciplineTime1132))
772     TeachingS["A20"]["2301113"][2]==0;
773     if(card(DisciplineTime1132 inter AvaiTimeA21)!=card(
            DisciplineTime1132))
```

```
774     TeachingS["A21"]["2301113"][2]==0;
775     if(card(DisciplineTime1132 inter AvaiTimeA22)!=card(
            DisciplineTime1132))
776     TeachingS["A22"]["2301113"][2]==0;
777     if(card(DisciplineTime1132 inter AvaiTimeA23)!=card(
            DisciplineTime1132))
778     TeachingS["A23"]["2301113"][2]==0;
779     if(card(DisciplineTime1132 inter AvaiTimeA31)!=card(
            DisciplineTime1132))
780     TeachingS["A31"]["2301113"][2]==0;
781     if(card(DisciplineTime1132 inter AvaiTimeA41)!=card(
            DisciplineTime1132))
782     TeachingS["A41"]["2301113"][2]==0;
783     if(card(DisciplineTime1132 inter AvaiTimeA42)!=card(
            DisciplineTime1132))
784     TeachingS["A42"]["2301113"][2]==0;
785     if(card(DisciplineTime1132 inter AvaiTimeA46)!=card(
            DisciplineTime1132))
786     TeachingS["A46"]["2301113"][2]==0;
787     if(card(DisciplineTime1132 inter AvaiTimeA48)!=card(
            DisciplineTime1132))
788     TeachingS["A48"]["2301113"][2]==0;
789     if(card(DisciplineTime1132 inter AvaiTimeA57)!=card(
            DisciplineTime1132))
790     TeachingS["A57"]["2301113"][2]==0;
791     // Available time for discipline 2301115 sec 1
792     if(card(DisciplineTime1151 inter AvaiTimeA13)!=card(
            DisciplineTime1151))
793     TeachingS["A13"]["2301115"][1]==0;
794     if(card(DisciplineTime1151 inter AvaiTimeA17)!=card(
            DisciplineTime1151))
795     TeachingS["A17"]["2301115"][1]==0;
```

```
796     if(card(DisciplineTime1151 inter AvaiTimeA19)!=card(
            DisciplineTime1151))
797     TeachingS["A19"]["2301115"][1]==0;
798     if(card(DisciplineTime1151 inter AvaiTimeA20)!=card(
            DisciplineTime1151))
799     TeachingS["A20"]["2301115"][1]==0;
800     if(card(DisciplineTime1151 inter AvaiTimeA21)!=card(
            DisciplineTime1151))
801     TeachingS["A21"]["2301115"][1]==0;
802     if(card(DisciplineTime1151 inter AvaiTimeA23)!=card(
            DisciplineTime1151))
803     TeachingS["A23"]["2301115"][1]==0;
804     if(card(DisciplineTime1151 inter AvaiTimeA27)!=card(
            DisciplineTime1151))
805     TeachingS["A27"]["2301115"][1]==0;
806     if(card(DisciplineTime1151 inter AvaiTimeA31)!=card(
            DisciplineTime1151))
807     TeachingS["A31"]["2301115"][1]==0;
808     if(card(DisciplineTime1151 inter AvaiTimeA32)!=card(
            DisciplineTime1151))
809     TeachingS["A32"]["2301115"][1]==0;
810     if(card(DisciplineTime1151 inter AvaiTimeA39)!=card(
            DisciplineTime1151))
811     TeachingS["A39"]["2301115"][1]==0;
812     if(card(DisciplineTime1151 inter AvaiTimeA41)!=card(
            DisciplineTime1151))
813     TeachingS["A41"]["2301115"][1]==0;
814     if(card(DisciplineTime1151 inter AvaiTimeA42)!=card(
            DisciplineTime1151))
815     TeachingS["A42"]["2301115"][1]==0;
816     // Available time for discipline 2301115 sec 2
817     if(card(DisciplineTime1152 inter AvaiTimeA13)!=card(
            DisciplineTime1152))
```

```
818     TeachingS["A13"]["2301115"][2]==0;
819     if(card(DisciplineTime1152 inter AvaiTimeA17)!=card(
            DisciplineTime1152))
820     TeachingS["A17"]["2301115"][2]==0;
821     if(card(DisciplineTime1152 inter AvaiTimeA19)!=card(
            DisciplineTime1152))
822     TeachingS["A19"]["2301115"][2]==0;
823     if(card(DisciplineTime1152 inter AvaiTimeA20)!=card(
            DisciplineTime1152))
824     TeachingS["A20"]["2301115"][2]==0;
825     if(card(DisciplineTime1152 inter AvaiTimeA21)!=card(
            DisciplineTime1152))
826     TeachingS["A21"]["2301115"][2]==0;
827     if(card(DisciplineTime1152 inter AvaiTimeA23)!=card(
            DisciplineTime1152))
828     TeachingS["A23"]["2301115"][2]==0;
829     if(card(DisciplineTime1152 inter AvaiTimeA27)!=card(
            DisciplineTime1152))
830     TeachingS["A27"]["2301115"][2]==0;
831     if(card(DisciplineTime1152 inter AvaiTimeA31)!=card(
            DisciplineTime1152))
832     TeachingS["A31"]["2301115"][2]==0;
833     if(card(DisciplineTime1152 inter AvaiTimeA32)!=card(
            DisciplineTime1152))
834     TeachingS["A32"]["2301115"][2]==0;
835     if(card(DisciplineTime1152 inter AvaiTimeA39)!=card(
            DisciplineTime1152))
836     TeachingS["A39"]["2301115"][2]==0;
837     if(card(DisciplineTime1152 inter AvaiTimeA41)!=card(
            DisciplineTime1152))
838     TeachingS["A41"]["2301115"][2]==0;
839     if(card(DisciplineTime1152 inter AvaiTimeA42)!=card(
            DisciplineTime1152))
```

```
840     TeachingS["A42"]["2301115"][2]==0;
841     // Available time for discipline 2301117 sec 1
842     if(card(DisciplineTime1171 inter AvaiTimeA6)!=card(
            DisciplineTime1171))
843     TeachingS["A6"]["2301117"][1]==0;
844     if(card(DisciplineTime1171 inter AvaiTimeA13)!=card(
            DisciplineTime1171))
845     TeachingS["A13"]["2301117"][1]==0;
846     if(card(DisciplineTime1171 inter AvaiTimeA15)!=card(
            DisciplineTime1171))
847     TeachingS["A15"]["2301117"][1]==0;
848     if(card(DisciplineTime1171 inter AvaiTimeA17)!=card(
            DisciplineTime1171))
849     TeachingS["A17"]["2301117"][1]==0;
850     if(card(DisciplineTime1171 inter AvaiTimeA18)!=card(
            DisciplineTime1171))
851     TeachingS["A18"]["2301117"][1]==0;
852     if(card(DisciplineTime1171 inter AvaiTimeA19)!=card(
            DisciplineTime1171))
853     TeachingS["A19"]["2301117"][1]==0;
854     if(card(DisciplineTime1171 inter AvaiTimeA20)!=card(
            DisciplineTime1171))
855     TeachingS["A20"]["2301117"][1]==0;
856     if(card(DisciplineTime1171 inter AvaiTimeA21)!=card(
            DisciplineTime1171))
857     TeachingS["A21"]["2301117"][1]==0;
858     if(card(DisciplineTime1171 inter AvaiTimeA23)!=card(
            DisciplineTime1171))
859     TeachingS["A23"]["2301117"][1]==0;
860     if(card(DisciplineTime1171 inter AvaiTimeA30)!=card(
            DisciplineTime1171))
861     TeachingS["A30"]["2301117"][1]==0;
```

```
862     if(card(DisciplineTime1171 inter AvaiTimeA31)!=card(
            DisciplineTime1171))
863     TeachingS["A31"]["2301117"][1]==0;
864     if(card(DisciplineTime1171 inter AvaiTimeA34)!=card(
            DisciplineTime1171))
865     TeachingS["A34"]["2301117"][1]==0;
866     if(card(DisciplineTime1171 inter AvaiTimeA39)!=card(
            DisciplineTime1171))
867     TeachingS["A39"]["2301117"][1]==0;
868     if(card(DisciplineTime1171 inter AvaiTimeA41)!=card(
            DisciplineTime1171))
869     TeachingS["A41"]["2301117"][1]==0;
870     if(card(DisciplineTime1171 inter AvaiTimeA42)!=card(
            DisciplineTime1171))
871     TeachingS["A42"]["2301117"][1]==0;
872     if(card(DisciplineTime1171 inter AvaiTimeA44)!=card(
            DisciplineTime1171))
873     TeachingS["A44"]["2301117"][1]==0;
874     if(card(DisciplineTime1171 inter AvaiTimeA45)!=card(
            DisciplineTime1171))
875     TeachingS["A45"]["2301117"][1]==0;
876     if(card(DisciplineTime1171 inter AvaiTimeA47)!=card(
            DisciplineTime1171))
877     TeachingS["A47"]["2301117"][1]==0;
878     if(card(DisciplineTime1171 inter AvaiTimeA48)!=card(
            DisciplineTime1171))
879     TeachingS["A48"]["2301117"][1]==0;
880     if(card(DisciplineTime1171 inter AvaiTimeA52)!=card(
            DisciplineTime1171))
881     TeachingS["A52"]["2301117"][1]==0;
882     if(card(DisciplineTime1171 inter AvaiTimeA54)!=card(
            DisciplineTime1171))
883     TeachingS["A54"]["2301117"][1]==0;
```

```
884     if(card(DisciplineTime1171 inter AvaiTimeA59)!=card(
            DisciplineTime1171))
885     TeachingS["A59"]["2301117"][1]==0;
886     // Available time for discipline 2301117 sec 2
887     if(card(DisciplineTime1172 inter AvaiTimeA6)!=card(
            DisciplineTime1172))
888     TeachingS["A6"]["2301117"][2]==0;
889     if(card(DisciplineTime1172 inter AvaiTimeA13)!=card(
            DisciplineTime1172))
890     TeachingS["A13"]["2301117"][2]==0;
891     if(card(DisciplineTime1172 inter AvaiTimeA15)!=card(
            DisciplineTime1172))
892     TeachingS["A15"]["2301117"][2]==0;
893     if(card(DisciplineTime1172 inter AvaiTimeA17)!=card(
            DisciplineTime1172))
894     TeachingS["A17"]["2301117"][2]==0;
895     if(card(DisciplineTime1172 inter AvaiTimeA18)!=card(
            DisciplineTime1172))
896     TeachingS["A18"]["2301117"][2]==0;
897     if(card(DisciplineTime1172 inter AvaiTimeA19)!=card(
            DisciplineTime1172))
898     TeachingS["A19"]["2301117"][2]==0;
899     if(card(DisciplineTime1172 inter AvaiTimeA20)!=card(
            DisciplineTime1172))
900     TeachingS["A20"]["2301117"][2]==0;
901     if(card(DisciplineTime1172 inter AvaiTimeA21)!=card(
            DisciplineTime1172))
902     TeachingS["A21"]["2301117"][2]==0;
903     if(card(DisciplineTime1172 inter AvaiTimeA23)!=card(
            DisciplineTime1172))
904     TeachingS["A23"]["2301117"][2]==0;
905     if(card(DisciplineTime1172 inter AvaiTimeA30)!=card(
            DisciplineTime1172))
```

```
906     TeachingS["A30"]["2301117"][2]==0;
907     if(card(DisciplineTime1172 inter AvaiTimeA31)!=card(
            DisciplineTime1172))
908     TeachingS["A31"]["2301117"][2]==0;
909     if(card(DisciplineTime1172 inter AvaiTimeA34)!=card(
            DisciplineTime1172))
910     TeachingS["A34"]["2301117"][2]==0;
911     if(card(DisciplineTime1172 inter AvaiTimeA39)!=card(
            DisciplineTime1172))
912     TeachingS["A39"]["2301117"][2]==0;
913     if(card(DisciplineTime1172 inter AvaiTimeA41)!=card(
            DisciplineTime1172))
914     TeachingS["A41"]["2301117"][2]==0;
915     if(card(DisciplineTime1172 inter AvaiTimeA42)!=card(
            DisciplineTime1172))
916     TeachingS["A42"]["2301117"][2]==0;
917     if(card(DisciplineTime1172 inter AvaiTimeA44)!=card(
            DisciplineTime1172))
918     TeachingS["A44"]["2301117"][2]==0;
919     if(card(DisciplineTime1172 inter AvaiTimeA45)!=card(
            DisciplineTime1172))
920     TeachingS["A45"]["2301117"][2]==0;
921     if(card(DisciplineTime1172 inter AvaiTimeA47)!=card(
            DisciplineTime1172))
922     TeachingS["A47"]["2301117"][2]==0;
923     if(card(DisciplineTime1172 inter AvaiTimeA48)!=card(
            DisciplineTime1172))
924     TeachingS["A48"]["2301117"][2]==0;
925     if(card(DisciplineTime1172 inter AvaiTimeA52)!=card(
            DisciplineTime1172))
926     TeachingS["A52"]["2301117"][2]==0;
927     if(card(DisciplineTime1172 inter AvaiTimeA54)!=card(
            DisciplineTime1172))
```

```
928     TeachingS["A54"]["2301117"][2]==0;
929     if(card(DisciplineTime1172 inter AvaiTimeA59)!=card(
            DisciplineTime1172))
930     TeachingS["A59"]["2301117"][2]==0;
931     // Available time for discipline 2301117 sec 3
932     if(card(DisciplineTime1173 inter AvaiTimeA6)!=card(
            DisciplineTime1173))
933     TeachingS["A6"]["2301117"][3]==0;
934     if(card(DisciplineTime1173 inter AvaiTimeA13)!=card(
            DisciplineTime1173))
935     TeachingS["A13"]["2301117"][3]==0;
936     if(card(DisciplineTime1173 inter AvaiTimeA15)!=card(
            DisciplineTime1173))
937     TeachingS["A15"]["2301117"][3]==0;
938     if(card(DisciplineTime1173 inter AvaiTimeA17)!=card(
            DisciplineTime1173))
939     TeachingS["A17"]["2301117"][3]==0;
940     if(card(DisciplineTime1173 inter AvaiTimeA18)!=card(
            DisciplineTime1173))
941     TeachingS["A18"]["2301117"][3]==0;
942     if(card(DisciplineTime1173 inter AvaiTimeA19)!=card(
            DisciplineTime1173))
943     TeachingS["A19"]["2301117"][3]==0;
944     if(card(DisciplineTime1173 inter AvaiTimeA20)!=card(
            DisciplineTime1173))
945     TeachingS["A20"]["2301117"][3]==0;
946     if(card(DisciplineTime1173 inter AvaiTimeA21)!=card(
            DisciplineTime1173))
947     TeachingS["A21"]["2301117"][3]==0;
948     if(card(DisciplineTime1173 inter AvaiTimeA23)!=card(
            DisciplineTime1173))
949     TeachingS["A23"]["2301117"][3]==0;
```

```
950     if(card(DisciplineTime1173 inter AvaiTimeA30)!=card(
            DisciplineTime1173))
951     TeachingS["A30"]["2301117"][3]==0;
952     if(card(DisciplineTime1173 inter AvaiTimeA31)!=card(
            DisciplineTime1173))
953     TeachingS["A31"]["2301117"][3]==0;
954     if(card(DisciplineTime1173 inter AvaiTimeA34)!=card(
            DisciplineTime1173))
955     TeachingS["A34"]["2301117"][3]==0;
956     if(card(DisciplineTime1173 inter AvaiTimeA39)!=card(
            DisciplineTime1173))
957     TeachingS["A39"]["2301117"][3]==0;
958     if(card(DisciplineTime1173 inter AvaiTimeA41)!=card(
            DisciplineTime1173))
959     TeachingS["A41"]["2301117"][3]==0;
960     if(card(DisciplineTime1173 inter AvaiTimeA42)!=card(
            DisciplineTime1173))
961     TeachingS["A42"]["2301117"][3]==0;
962     if(card(DisciplineTime1173 inter AvaiTimeA44)!=card(
            DisciplineTime1173))
963     TeachingS["A44"]["2301117"][3]==0;
964     if(card(DisciplineTime1173 inter AvaiTimeA45)!=card(
            DisciplineTime1173))
965     TeachingS["A45"]["2301117"][3]==0;
966     if(card(DisciplineTime1173 inter AvaiTimeA47)!=card(
            DisciplineTime1173))
967     TeachingS["A47"]["2301117"][3]==0;
968     if(card(DisciplineTime1173 inter AvaiTimeA48)!=card(
            DisciplineTime1173))
969     TeachingS["A48"]["2301117"][3]==0;
970     if(card(DisciplineTime1173 inter AvaiTimeA52)!=card(
            DisciplineTime1173))
971     TeachingS["A52"]["2301117"][3]==0;
```

```
972     if(card(DisciplineTime1173 inter AvaiTimeA54)!=card(
            DisciplineTime1173))
973     TeachingS["A54"]["2301117"][3]==0;
974     if(card(DisciplineTime1173 inter AvaiTimeA59)!=card(
            DisciplineTime1173))
975     TeachingS["A59"]["2301117"][3]==0;
976     // Available time for discipline 2301117 sec 4
977     if(card(DisciplineTime1174 inter AvaiTimeA6)!=card(
            DisciplineTime1174))
978     TeachingS["A6"]["2301117"][4]==0;
979     if(card(DisciplineTime1174 inter AvaiTimeA13)!=card(
            DisciplineTime1174))
980     TeachingS["A13"]["2301117"][4]==0;
981     if(card(DisciplineTime1174 inter AvaiTimeA15)!=card(
            DisciplineTime1174))
982     TeachingS["A15"]["2301117"][4]==0;
983     if(card(DisciplineTime1174 inter AvaiTimeA17)!=card(
            DisciplineTime1174))
984     TeachingS["A17"]["2301117"][4]==0;
985     if(card(DisciplineTime1174 inter AvaiTimeA18)!=card(
            DisciplineTime1174))
986     TeachingS["A18"]["2301117"][4]==0;
987     if(card(DisciplineTime1174 inter AvaiTimeA19)!=card(
            DisciplineTime1174))
988     TeachingS["A19"]["2301117"][4]==0;
989     if(card(DisciplineTime1174 inter AvaiTimeA20)!=card(
            DisciplineTime1174))
990     TeachingS["A20"]["2301117"][4]==0;
991     if(card(DisciplineTime1174 inter AvaiTimeA21)!=card(
            DisciplineTime1174))
992     TeachingS["A21"]["2301117"][4]==0;
993     if(card(DisciplineTime1174 inter AvaiTimeA23)!=card(
            DisciplineTime1174))
```

```
994     TeachingS["A23"]["2301117"][4]==0;
995     if(card(DisciplineTime1174 inter AvaiTimeA30)!=card(
            DisciplineTime1174))
996     TeachingS["A30"]["2301117"][4]==0;
997     if(card(DisciplineTime1174 inter AvaiTimeA31)!=card(
            DisciplineTime1174))
998     TeachingS["A31"]["2301117"][4]==0;
999     if(card(DisciplineTime1174 inter AvaiTimeA34)!=card(
            DisciplineTime1174))
1000    TeachingS["A34"]["2301117"][4]==0;
1001    if(card(DisciplineTime1174 inter AvaiTimeA39)!=card(
            DisciplineTime1174))
1002    TeachingS["A39"]["2301117"][4]==0;
1003    if(card(DisciplineTime1174 inter AvaiTimeA41)!=card(
            DisciplineTime1174))
1004    TeachingS["A41"]["2301117"][4]==0;
1005    if(card(DisciplineTime1174 inter AvaiTimeA42)!=card(
            DisciplineTime1174))
1006    TeachingS["A42"]["2301117"][4]==0;
1007    if(card(DisciplineTime1174 inter AvaiTimeA44)!=card(
            DisciplineTime1174))
1008    TeachingS["A44"]["2301117"][4]==0;
1009    if(card(DisciplineTime1174 inter AvaiTimeA45)!=card(
            DisciplineTime1174))
1010    TeachingS["A45"]["2301117"][4]==0;
1011    if(card(DisciplineTime1174 inter AvaiTimeA47)!=card(
            DisciplineTime1174))
1012    TeachingS["A47"]["2301117"][4]==0;
1013    if(card(DisciplineTime1174 inter AvaiTimeA48)!=card(
            DisciplineTime1174))
1014    TeachingS["A48"]["2301117"][4]==0;
1015    if(card(DisciplineTime1174 inter AvaiTimeA52)!=card(
            DisciplineTime1174))
```

```
1016    TeachingS["A52"]["2301117"][4]==0;
1017    if(card(DisciplineTime1174 inter AvaiTimeA54)!=card(
            DisciplineTime1174))
1018    TeachingS["A54"]["2301117"][4]==0;
1019    if(card(DisciplineTime1174 inter AvaiTimeA59)!=card(
            DisciplineTime1174))
1020    TeachingS["A59"]["2301117"][4]==0;
1021    // Available time for discipline 2301119 sec 1
1022    if(card(DisciplineTime1191 inter AvaiTimeA17)!=card(
            DisciplineTime1191))
1023    TeachingS["A17"]["2301119"][1]==0;
1024    if(card(DisciplineTime1191 inter AvaiTimeA19)!=card(
            DisciplineTime1191))
1025    TeachingS["A19"]["2301119"][1]==0;
1026    if(card(DisciplineTime1191 inter AvaiTimeA20)!=card(
            DisciplineTime1191))
1027    TeachingS["A20"]["2301119"][1]==0;
1028    if(card(DisciplineTime1191 inter AvaiTimeA21)!=card(
            DisciplineTime1191))
1029    TeachingS["A21"]["2301119"][1]==0;
1030    if(card(DisciplineTime1191 inter AvaiTimeA23)!=card(
            DisciplineTime1191))
1031    TeachingS["A23"]["2301119"][1]==0;
1032    if(card(DisciplineTime1191 inter AvaiTimeA27)!=card(
            DisciplineTime1191))
1033    TeachingS["A27"]["2301119"][1]==0;
1034    if(card(DisciplineTime1191 inter AvaiTimeA31)!=card(
            DisciplineTime1191))
1035    TeachingS["A31"]["2301119"][1]==0;
1036    if(card(DisciplineTime1191 inter AvaiTimeA41)!=card(
            DisciplineTime1191))
1037    TeachingS["A41"]["2301119"][1]==0;
```

```
1038    if(card(DisciplineTime1191 inter AvaiTimeA42)!=card(
            DisciplineTime1191))
1039    TeachingS["A42"]["2301119"][1]==0;
1040    // Available time for discipline 2301119 sec 2
1041    if(card(DisciplineTime1192 inter AvaiTimeA17)!=card(
            DisciplineTime1192))
1042    TeachingS["A17"]["2301119"][2]==0;
1043    if(card(DisciplineTime1192 inter AvaiTimeA19)!=card(
            DisciplineTime1192))
1044    TeachingS["A19"]["2301119"][2]==0;
1045    if(card(DisciplineTime1192 inter AvaiTimeA20)!=card(
            DisciplineTime1192))
1046    TeachingS["A20"]["2301119"][2]==0;
1047    if(card(DisciplineTime1192 inter AvaiTimeA21)!=card(
            DisciplineTime1192))
1048    TeachingS["A21"]["2301119"][2]==0;
1049    if(card(DisciplineTime1192 inter AvaiTimeA23)!=card(
            DisciplineTime1192))
1050    TeachingS["A23"]["2301119"][2]==0;
1051    if(card(DisciplineTime1192 inter AvaiTimeA27)!=card(
            DisciplineTime1192))
1052    TeachingS["A27"]["2301119"][2]==0;
1053    if(card(DisciplineTime1192 inter AvaiTimeA31)!=card(
            DisciplineTime1192))
1054    TeachingS["A31"]["2301119"][2]==0;
1055    if(card(DisciplineTime1192 inter AvaiTimeA41)!=card(
            DisciplineTime1192))
1056    TeachingS["A41"]["2301119"][2]==0;
1057    if(card(DisciplineTime1192 inter AvaiTimeA42)!=card(
            DisciplineTime1192))
1058    TeachingS["A42"]["2301119"][2]==0;
1059    // Available time for discipline 2301119 sec 3
```

```
1060    if(card(DisciplineTime1193 inter AvaiTimeA17)!=card(
            DisciplineTime1193))
1061    TeachingS["A17"]["2301119"][3]==0;
1062    if(card(DisciplineTime1193 inter AvaiTimeA19)!=card(
            DisciplineTime1193))
1063    TeachingS["A19"]["2301119"][3]==0;
1064    if(card(DisciplineTime1193 inter AvaiTimeA20)!=card(
            DisciplineTime1193))
1065    TeachingS["A20"]["2301119"][3]==0;
1066    if(card(DisciplineTime1193 inter AvaiTimeA21)!=card(
            DisciplineTime1193))
1067    TeachingS["A21"]["2301119"][3]==0;
1068    if(card(DisciplineTime1193 inter AvaiTimeA23)!=card(
            DisciplineTime1193))
1069    TeachingS["A23"]["2301119"][3]==0;
1070    if(card(DisciplineTime1193 inter AvaiTimeA27)!=card(
            DisciplineTime1193))
1071    TeachingS["A27"]["2301119"][3]==0;
1072    if(card(DisciplineTime1193 inter AvaiTimeA31)!=card(
            DisciplineTime1193))
1073    TeachingS["A31"]["2301119"][3]==0;
1074    if(card(DisciplineTime1193 inter AvaiTimeA41)!=card(
            DisciplineTime1193))
1075    TeachingS["A41"]["2301119"][3]==0;
1076    if(card(DisciplineTime1193 inter AvaiTimeA42)!=card(
            DisciplineTime1193))
1077    TeachingS["A42"]["2301119"][3]==0;
1078
1079    sum(i in Teacher, t in DisciplineTime1011) TeachingST[i]["2301101"
            ][1][t]==card(DisciplineTime1011);
1080    sum(i in Teacher, t in DisciplineTime1012) TeachingST[i]["2301101"
            ][2][t]==card(DisciplineTime1012);
1081
```

```
1082    sum(i in Teacher, t in DisciplineTime1031) TeachingST[i]["2301103"
            ][1][t]==card(DisciplineTime1031);
1083    sum(i in Teacher, t in DisciplineTime1032) TeachingST[i]["2301103"
            ][2][t]==card(DisciplineTime1032);
1084    sum(i in Teacher, t in DisciplineTime1033) TeachingST[i]["2301103"
            ][3][t]==card(DisciplineTime1033);
1085
1086    sum(i in Teacher, t in DisciplineTime1071) TeachingST[i]["2301107"
            ][1][t]==card(DisciplineTime1071);
1087    sum(i in Teacher, t in DisciplineTime1072) TeachingST[i]["2301107"
            ][2][t]==card(DisciplineTime1072);
1088    sum(i in Teacher, t in DisciplineTime1073) TeachingST[i]["2301107"
            ][3][t]==card(DisciplineTime1073);
1089    sum(i in Teacher, t in DisciplineTime1074) TeachingST[i]["2301107"
            ][4][t]==card(DisciplineTime1074);
1090    sum(i in Teacher, t in DisciplineTime1075) TeachingST[i]["2301107"
            ][5][t]==card(DisciplineTime1075);
1091    sum(i in Teacher, t in DisciplineTime1076) TeachingST[i]["2301107"
            ][6][t]==card(DisciplineTime1076);
1092    sum(i in Teacher, t in DisciplineTime1077) TeachingST[i]["2301107"
            ][7][t]==card(DisciplineTime1077);
1093
1094    sum(i in Teacher, t in DisciplineTime1131) TeachingST[i]["2301113"
            ][1][t]==card(DisciplineTime1131);
1095    sum(i in Teacher, t in DisciplineTime1132) TeachingST[i]["2301113"
            ][2][t]==card(DisciplineTime1132);
1096
1097    sum(i in Teacher, t in DisciplineTime1151) TeachingST[i]["2301115"
            ][1][t]==card(DisciplineTime1151);
1098    sum(i in Teacher, t in DisciplineTime1152) TeachingST[i]["2301115"
            ][2][t]==card(DisciplineTime1152);
1099
```

```
1100    sum(i in Teacher, t in DisciplineTime1171) TeachingST[i]["2301117"
            ][1][t]==card(DisciplineTime1171);
1101    sum(i in Teacher, t in DisciplineTime1172) TeachingST[i]["2301117"
            ][2][t]==card(DisciplineTime1172);
1102    sum(i in Teacher, t in DisciplineTime1173) TeachingST[i]["2301117"
            ][3][t]==card(DisciplineTime1173);
1103    sum(i in Teacher, t in DisciplineTime1174) TeachingST[i]["2301117"
            ][4][t]==card(DisciplineTime1174);
1104
1105    sum(i in Teacher, t in DisciplineTime1191) TeachingST[i]["2301119"
            ][1][t]==card(DisciplineTime1191);
1106    sum(i in Teacher, t in DisciplineTime1192) TeachingST[i]["2301119"
            ][2][t]==card(DisciplineTime1192);
1107    sum(i in Teacher, t in DisciplineTime1193) TeachingST[i]["2301119"
            ][3][t]==card(DisciplineTime1193);
1108
1109    sum(i in Teacher, t in DisciplineTime6751) TeachingST[i]["2301675"
            ][1][t]==card(DisciplineTime6751);
1110    // Available time for discipline 2301675 sec 1
1111    if(card(DisciplineTime6751 inter AvaiTimeA4)!=card(
            DisciplineTime6751))
1112    TeachingS["A4"]["2301675"][1]==0;
1113    if(card(DisciplineTime6751 inter AvaiTimeA33)!=card(
            DisciplineTime6751))
1114    TeachingS["A33"]["2301675"][1]==0;
1115    //Each instructor can only teach at most one section for each course
            .
1116    forall(i in Teacher, j in Discipline)
1117    sum(k in Section) TeachingS[i][j][k] <= 1;
1118    //An instructor i allow to teach at most one course section at any
            time slot.
1119    forall(i in Teacher, t in TimeSlot)
1120    sum(j in Discipline, k in Section) TeachingST[i][j][k][t] <= 1;
```

```
1121    //An overload or underload of an instructor is his/her total
            assigned workload subtracting with the requested workload.
1122    forall(i in Teacher)
1123    PenaltyWorkload[i] == (sum(j in Discipline) DisciplineWorkload[j]/2*
            Teaching[i][j] + OldWorkload[i]) - RequestedWorkload[i];
1124    //An instructor i must be assigned to a course j first before be
            able to know the course section k.
1125    forall(i in Teacher, j in Discipline, k in Section)
1126    Teaching[i][j] >= TeachingS[i][j][k];
1127    //An instructor i must be assigned to the section k of course j
            before choosing the time slot $t$.
1128    forall(i in Teacher, j in Discipline, k in Section, t in TimeSlot){
1129    TeachingS[i][j][k] >= TeachingST[i][j][k][t]; }
1130    }
```

The second stage model.dat

```
SheetConnection sheet("Data-for-the-second-step-model.xlsx");
Discipline from SheetRead(sheet,"Data!A2:A9");
Teacher101 from SheetRead(sheet,"Data!D2:D18");
Teacher103 from SheetRead(sheet,"Data!D19:D32");
Teacher107 from SheetRead(sheet,"Data!D33:D59");
Teacher113 from SheetRead(sheet,"Data!D60:D71");
Teacher115 from SheetRead(sheet,"Data!D72:D83");
Teacher117 from SheetRead(sheet,"Data!D84:D105");
Teacher119 from SheetRead(sheet,"Data!D106:D114");
Teacher675 from SheetRead(sheet,"Data!D115:D116");

Teacher from SheetRead(sheet,"Workload!A2:A61");
TimeSlot from SheetRead(sheet,"Sheet12!H2:H81");

AvaiTimeA1 from SheetRead(sheet,"Sheet12!U2:U81");
AvaiTimeA2 from SheetRead(sheet,"Sheet12!BK2:BK81");
AvaiTimeA3 from SheetRead(sheet,"Sheet12!R2:R81");
AvaiTimeA4 from SheetRead(sheet,"Sheet12!BJ2:BJ81");
AvaiTimeA5 from SheetRead(sheet,"Sheet12!T2:T81");
AvaiTimeA6 from SheetRead(sheet,"Sheet12!S2:S81");
AvaiTimeA7 from SheetRead(sheet,"Sheet12!BI2:BI81");
AvaiTimeA8 from SheetRead(sheet,"Sheet12!M2:M81");
AvaiTimeA9 from SheetRead(sheet,"Sheet12!P2:P81");
AvaiTimeA10 from SheetRead(sheet,"Sheet12!Q2:Q81");
AvaiTimeA11 from SheetRead(sheet,"Sheet12!BH2:BH81");
AvaiTimeA12 from SheetRead(sheet,"Sheet12!BG2:BG81");
AvaiTimeA13 from SheetRead(sheet,"Sheet12!N2:N81");
AvaiTimeA14 from SheetRead(sheet,"Sheet12!O2:O81");
AvaiTimeA15 from SheetRead(sheet,"Sheet12!Z2:Z81");
AvaiTimeA16 from SheetRead(sheet,"Sheet12!X2:X81");
AvaiTimeA17 from SheetRead(sheet,"Sheet12!Y2:Y81");
```

```
32    AvaiTimeA18 from SheetRead(sheet,"Sheet12!AX2:AX81");
33    AvaiTimeA19 from SheetRead(sheet,"Sheet12!AQ2:AQ81");
34    AvaiTimeA20 from SheetRead(sheet,"Sheet12!AW2:AW81");
35    AvaiTimeA21 from SheetRead(sheet,"Sheet12!AV2:AV81");
36    AvaiTimeA22 from SheetRead(sheet,"Sheet12!AU2:AU81");
37    AvaiTimeA23 from SheetRead(sheet,"Sheet12!AT2:AT81");
38    AvaiTimeA24 from SheetRead(sheet,"Sheet12!BL2:BL81");
39    AvaiTimeA25 from SheetRead(sheet,"Sheet12!W2:W81");
40    AvaiTimeA26 from SheetRead(sheet,"Sheet12!BF2:BF81");
41    AvaiTimeA27 from SheetRead(sheet,"Sheet12!AC2:AC81");
42    AvaiTimeA28 from SheetRead(sheet,"Sheet12!AB2:AB81");
43    AvaiTimeA29 from SheetRead(sheet,"Sheet12!BN2:BN81");
44    AvaiTimeA30 from SheetRead(sheet,"Sheet12!AH2:AH81");
45    AvaiTimeA31 from SheetRead(sheet,"Sheet12!AG2:AG81");
46    AvaiTimeA32 from SheetRead(sheet,"Sheet12!AF2:AF81");
47    AvaiTimeA33 from SheetRead(sheet,"Sheet12!AE2:AE81");
48    AvaiTimeA34 from SheetRead(sheet,"Sheet12!AI2:AI81");
49    AvaiTimeA35 from SheetRead(sheet,"Sheet12!AD2:AD81");
50    AvaiTimeA36 from SheetRead(sheet,"Sheet12!BM2:BM81");
51    AvaiTimeA37 from SheetRead(sheet,"Sheet12!V2:V81");
52    AvaiTimeA38 from SheetRead(sheet,"Sheet12!BD2:BD81");
53    AvaiTimeA39 from SheetRead(sheet,"Sheet12!BE2:BE81");
54    AvaiTimeA40 from SheetRead(sheet,"Sheet12!AJ2:AJ81");
55    AvaiTimeA41 from SheetRead(sheet,"Sheet12!AK2:AK81");
56    AvaiTimeA42 from SheetRead(sheet,"Sheet12!AL2:AL81");
57    AvaiTimeA43 from SheetRead(sheet,"Sheet12!BB2:BB81");
58    AvaiTimeA44 from SheetRead(sheet,"Sheet12!BA2:BA81");
59    AvaiTimeA45 from SheetRead(sheet,"Sheet12!AZ2:AZ81");
60    AvaiTimeA46 from SheetRead(sheet,"Sheet12!AY2:AY81");
61    AvaiTimeA47 from SheetRead(sheet,"Sheet12!BC2:BC81");
62    AvaiTimeA48 from SheetRead(sheet,"Sheet12!AM2:AM81");
63    AvaiTimeA49 from SheetRead(sheet,"Sheet12!AO2:AO81");
64    AvaiTimeA50 from SheetRead(sheet,"Sheet12!BO2:BO81");
```

```
65    AvaiTimeA51 from SheetRead(sheet,"Sheet12!BP2:BP81");
66    AvaiTimeA52 from SheetRead(sheet,"Sheet12!AS2:AS81");
67    AvaiTimeA53 from SheetRead(sheet,"Sheet12!AP2:AP81");
68    AvaiTimeA54 from SheetRead(sheet,"Sheet12!AN2:AN81");
69    AvaiTimeA55 from SheetRead(sheet,"Sheet12!AR2:AR81");
70    AvaiTimeA56 from SheetRead(sheet,"sheet12!I2:I81");
71    AvaiTimeA57 from SheetRead(sheet,"Sheet12!AA2:AA81");
72    AvaiTimeA58 from SheetRead(sheet,"Sheet12!K2:K81");
73    AvaiTimeA59 from SheetRead(sheet,"Sheet12!L2:L81");
74    AvaiTimeA60 from SheetRead(sheet,"Sheet12!J2:J81");
75
76    DisciplineTime1011 from SheetRead(sheet,"ResultToData!G2:G9");
77    DisciplineTime1012 from SheetRead(sheet,"ResultToData!G10:G17");
78
79    DisciplineTime1031 from SheetRead(sheet,"ResultToData!G18:G23");
80    DisciplineTime1032 from SheetRead(sheet,"ResultToData!G24:G29");
81    DisciplineTime1033 from SheetRead(sheet,"ResultToData!G30:G37");
82
83    DisciplineTime1071 from SheetRead(sheet,"ResultToData!G38:G43");
84    DisciplineTime1072 from SheetRead(sheet,"ResultToData!G44:G49");
85    DisciplineTime1073 from SheetRead(sheet,"ResultToData!G50:G55");
86    DisciplineTime1074 from SheetRead(sheet,"ResultToData!G56:G61");
87    DisciplineTime1075 from SheetRead(sheet,"ResultToData!G62:G67");
88    DisciplineTime1076 from SheetRead(sheet,"ResultToData!G68:G73");
89    DisciplineTime1077 from SheetRead(sheet,"ResultToData!G74:G79");
90
91    DisciplineTime1131 from SheetRead(sheet,"ResultToData!G80:G87");
92    DisciplineTime1132 from SheetRead(sheet,"ResultToData!G88:G95");
93
94    DisciplineTime1151 from SheetRead(sheet,"ResultToData!G96:G101");
95    DisciplineTime1152 from SheetRead(sheet,"ResultToData!G102:G107");
96
97    DisciplineTime1171 from SheetRead(sheet,"ResultToData!G124:G132");
```

```
98    DisciplineTime1172 from SheetRead(sheet,"ResultToData!G132:G139");
99    DisciplineTime1173 from SheetRead(sheet,"ResultToData!G140:G147");
100   DisciplineTime1174 from SheetRead(sheet,"ResultToData!G148:G155");
101
102   DisciplineTime1191 from SheetRead(sheet,"ResultToData!G156:G163");
103   DisciplineTime1192 from SheetRead(sheet,"ResultToData!G164:G171");
104   DisciplineTime1193 from SheetRead(sheet,"ResultToData!G172:G179");
105
106   DisciplineTime6751 from SheetRead(sheet,"ResultToData!G764:G771");
107
108   DSectionSet from SheetRead(sheet,"ResultToData!I2:K187");
109   TWorkloadSet from SheetRead(sheet,"Workload!A2:C61");
110   DWorkloadSet from SheetRead(sheet,"Workload!E2:F9");
111   TDPreferenceSet from SheetRead(sheet,"Data!D2:F116");
112   }
```

BIOGRAPHY

Name	Miss Nipitta Burana
Date of Birth	08 April 1996
Place of Birth	Chanthaburi, Thailand
Education	B.Sc. (Mathematics and Computer Science), Chulalongkorn University, 2017