COURSE SCHEDULER AND RECOMMENDATION SYSTEM FOR STUDENTS

by

Süleyman Uslu

B.S., Computer Engineering, Boğaziçi University, 2011

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2017

COURSE SCHEDULER AND RECOMMENDATION
SYSTEM FOR STUDENTS

APPROVED BY:

Prof. Can Özturan .....
(Thesis Supervisor)

Prof. Tunga Güngör .....

Asst. Prof. Reyhan Aydoğan .....

DATE OF APPROVAL: 1.7.2017

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Prof. Can Özturan for his guidance during this research. Without his guidance and persistent help, this study would not have been possible.

I also would like to thank Prof. Tunga Güngör and Asst. Prof. Reyhan Aydoğan for their participation in my thesis jury and their priceless comments.

Last but not least, I would like to thank my family. First, I would like to thank my brother for his contribution to the development of both front-end and optimization parts of my thesis. His knowledge of client-side programming and optimization techniques, and also his taste of aesthetics made this thesis possible. Then, I would like to thank my father and my mother for always supporting me. Lastly, I would like to thank my beloved wife for always being there for me.

ABSTRACT

COURSE SCHEDULER AND RECOMMENDATION SYSTEM FOR STUDENTS

University students need to prepare their course programs at the beginning of each semester. A large number of offered courses is desired by students since it is easier to create a schedule using a great diversity of courses. However, increase in the number of offered courses makes students to be indecisive about selecting courses. Although this problem exists for smaller number of courses, it becomes harder as the number of courses increases.

Another problem for students is that they may not be aware of some courses which are good fit for their preference and course program. They can make use of a course recommendation tool which helps them find their prospective courses.

Furthermore, students are also interested in making their graduation plan. They might have completed some courses and could be interested in arrangement of remaining ones into semesters.

In order to address these problems, Mecanin, an online platform for Boğaziçi University students that provides a course schedule optimizer, a course recommendation tool, and a graduation planner, is developed. In this thesis, integer programming models for course program optimization and graduation planning are proposed along with recommendation algorithms to suggest courses to students. Proposed algorithms are tested and compared with three baselines which are random, popular, and popularity-based probabilistic recommendation algorithms, and the results are presented.

ÖZET

ÖĞRENCİLER İÇİN DERS PROGRAMI ÇİZELGELEYİCİSİ VE DERS TAVSİYE SİSTEMİ

Her dönemin başında üniversite öğrencilerinin ders programlarına karar vermeleri gerekir. Çok sayıda ders alternatifi içinden programların hazırlanması daha kolay olacağı için öğrenciler daha çok ders sunulmasını ister. Ancak sunulan ders sayısının artması öğrencilerin ders seçimi konusunda kararsızlık yaşamasına sebep olur. Bu problem her ne kadar az ders adedinde de geçerli olsa da ders sayısı arttıkça problemin çözümü güçleşir.

Bir başka problem ise öğrenciler almak isteyebilecekleri ve programlarına uyan derslerin varlığından haberdar olmayabilirler. Kendilerine ders tavsiyesinde bulunabilecek bir sistem, bu açıdan faydalarına olabilir.

Ayrıca, ögrenciler mezuniyet planlarını yapmak isteyebilirler. Mezuniyet için gereken dersleri ve onların dönemlere nasıl paylaştırılacağını görmek isteyebilirler.

Bu problemlerin çözümü adına Boğaziçi Üniversitesi öğrencileri için tasarlanan, ders programını optimize eden, ders tavsiyesinde bulunan ve mezuniyet planlaması yapan bir online platform olan Mecanin geliştirilmiştir. Bu tezde, Mecanin'e girilen ders programlarını optimize etmekte kullanılan ve mezuniyet planı çıkaran iki ayrı tamsayı programlama modeli ve ders tavsiyesi algoritması sunulmuştur. Geliştirilen algoritmalar test edilmiş, rastgele, popülerlik tabanlı ve popülerlik tabanlı olasılıksal üç algoritmayla karşılaştırılmış ve karşılaştırma sonuçları gösterilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . x
LIST OF SYMBOLS . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
1.1. Motivation . . . 1
1.2. Contributions of the Thesis . . . 1
1.3. Outline . . . 2
2. BACKGROUND . . . 4
2.1. Linear Programming . . . 4
2.2. Recommendation Systems . . . 5
3. PREVIOUS WORK . . . 7
4. MECANIN SYSTEM ARCHITECTURE . . . 10
4.1. Web User Interface . . . 10
4.1.1. Registration and Login Page . . . 11
4.1.2. Profile Page . . . 11
4.1.3. Editor Page . . . 13
4.1.4. Course Pages . . . 13
4.1.5. User Pages . . . 14
4.2. Detailed Usage . . . 16
4.2.1. Grouping . . . 17
4.2.2. Ratings . . . 18
4.2.3. Additional Course Hours . . . 18
4.3. Recommendation . . . 19
4.4. Concepts . . . 19
4.4.1. Course Names . . . 20

4.4.2. Course Sections 20
4.4.3. Additional Hours 20
5. COURSE PROGRAM SCHEDULING ALGORITHMS 21
5.1. IP Model 21
5.2. Example 24
6. GRADUATION PLANNING ALGORITHMS 27
6.1. Usage 27
6.2. Concepts 28
6.3. IP Model 29
7. COURSE RECOMMENDATION ALGORITHMS 36
8. TESTS AND RESULTS 40
8.1. Mecanin Scheduler 40
8.2. Recommendation Engine 42
8.3. Statistics 43
8.4. Test Methods 43
8.5. Test Results 44
9. CONCLUSIONS 45
9.1. Future Work 45
REFERENCES 47

LIST OF FIGURES

Figure 4.1. Mecanin System Architecture 11

Figure 4.2. Mecanin registration and login page 12

Figure 4.3. Mecanin profile page 12

Figure 4.4. Mecanin editor page 14

Figure 4.5. Course page design of Mecanin 15

Figure 4.6. Other users' page in Mecanin 15

Figure 4.7. Mecanin wants users to confirm their course removals 17

Figure 4.8. An example of grouping in Mecanin 18

Figure 4.9. Star rating system to specify the importance of a course in Mecanin 19

Figure 4.10. Lab and PS hours of a course 19

Figure 5.1. Suggested course program by Mecanin Scheduler 25

Figure 5.2. Mecanin Scheduler with course preferences 26

Figure 5.3. Mecanin Scheduler with conflicts disabled 26

Figure 6.1. Interface of Mecanin graduation planner 28

Figure 6.2. Mecanin graduation planner with Summer disabled and 20 credits limit 30

Figure 7.1. Mecanin Course Recommendation Algorithm 37

Figure 7.2. Graph representation of course selections 38

Figure 8.1. Comparison of solving times 42

LIST OF TABLES

Table 8.1. Solution times in milliseconds 41

Table 8.2. Student and course statistics over 9 semesters 43

Table 8.3. Comparison of recommendation algorithms (ratio of successful course recommendations) 44

LIST OF SYMBOLS

a_i	Minimum number of courses to be selected in course group i
b_i	Maximum number of courses to be selected in course group i
c_i	Importance of course i
c_s	Weight of semester s
e	Node of G as a student selecting w
g	Number of groups
G	Course selection graph
n	Number of courses
n_i	Credit of course i
n_s	Number of selected courses
t_i	Total number of courses in course group i
v	Node of G as a student to get course recommendation
w	Node of G as a course of v
x	Node of G as a course of e
x_a^s	Decision variable of course a in semester s
x_i	Decision variable of course i

LIST OF ACRONYMS/ABBREVIATIONS

ANSI	American National Standards Institute
BIP	Binary Integer Programming
CC	Complementary Course
CF	Collaborative Filtering
CMPE	Computer Engineering Department
DAO	Database Access Object
DB	Database
DBMS	Database Management System
HSS	Humanities and Social Sciences
ILP	Integer Linear Programming
IP	Integer Programming
MDP	Markov Decision Process
POP	Popularity based course recommendation algorithm
RND	Random course recommendation algorithm
RWH	Roulette Wheel based course recommendation algorithm
WWII	World War II

1. INTRODUCTION

1.1. Motivation

Optimization and scheduling are important aspects of many processes from timing work shifts to assigning jobs to machines. As the number of offered courses increases, students start considering different courses as their elective courses, and this problem draws interest of researchers in the optimization field. As the most relevant example, graduate or senior undergraduate students need to decide optional courses while preparing their course programs.

There are many factors which affect the students' decision such as different times of courses, professor and course preferences, and conflicting hours of courses. It becomes difficult to handle these constraints by hand or mind when the number of available courses reaches hundreds.

In addition, students may not be aware of all the courses they can take. It also requires effort to find the best matching ones when there are many course alternatives. Sometimes, there may be courses which they have not heard of and also could be a good fit for their schedules.

Another difficulty that university students face is that they would like to know their semester plans for their remaining courses to see the time of their graduation. Once they try to arrange their program, they realize that constraints and course alternatives make the process very difficult and time consuming when it is done by hand.

1.2. Contributions of the Thesis

For university students, (i) an online platform, named Mecanin, is built to help them prepare their schedules considering their course and professor preferences as well as the time conflicts of courses. In addition, (ii) an integer programming (IP) model

is developed to find the optimum schedule for students and this tool is made available online as a part of Mecanin platform. Also, (iii) a collaborative filtering (CF) based recommendation algorithm is provided to help students find new courses which they might be interested in based on their and other students' course preferences. Several adjustments that are specific to course program planning have been considered to improve the success of the algorithm.

Moreover, to help students plan their graduation, (iv) a graduation planner tool is designed and implemented in Mecanin. Several constraints such as the offered courses per semesters, semester course and credit limits, and the curriculum of the department are considered, and a new IP model is proposed. Also, a web interface is designed for students to use this tool.

Mecanin online platform is used by Boğaziçi University students. The number of students that have used the system is between 700 and 1400 per semester for the past 9 consecutive semesters from 2010 to 2014. Algorithms are tested on this anonymized real dataset.

1.3. Outline

The thesis is organized as follows: In Chapter 2, some background information is given about integer programming and recommendation systems. In Chapter 3, related work for both optimization and recommendation problem is presented. In Chapter 4, the architecture and the usage of the Mecanin system is presented in detail. In Chapter 5, integer programming formulation is presented for scheduling the student's courses by providing the objective function and constraints. In Chapter 6, an IP model is presented for finding course plans of the semesters for the remaining courses that will lead to the earliest graduation. In Chapter 7, course recommendation algorithms are presented. Here, we describe the adaptation of CF into course recommendation algorithm and show the different variations of the this method. In Chapter 8, we first compare the running times of our IP solver implementation with an open source linear programming framework LPSolve and present the results of both implementations. Later, we

describe how the results of the recommendation algorithm are evaluated. We also show the results and compare the proposed methods with the basic techniques which are not based on similarity. Finally, Chapter 9 concludes the thesis with a discussion of results and possible future work.

2. BACKGROUND

This chapter provides preliminary information on linear programming and recommendation systems.

2.1. Linear Programming

Linear programming (LP) is a method to solve linear minimization or maximization problems in the existence of linear constraints [1].

During World War II (WWII), it was crucial to plan the military operations and the management of army sources, and planning even small number of sources was impossible to calculate because of the number of permutations. In 1947, George B. Dantzig developed Simplex method [2] to provide a very fast solution for linear problems by only checking very limited number of possibilities. After WWII, linear programming and Simplex method became widely used in many industrial fields and optimization problems [1].

A linear problem consists of two parts which are an objective function and constraints both of which have to be linear. Objective function can either be a maximization or minimization function, and there can be more than one constraint which can be either an equality or inequality [1].

If a solution satisfies all constraints, it is called a feasible solution, and the region that covers all feasible solutions is called feasible region. If the solution set is empty, in other words, if there is no feasible solution, the linear problem is called infeasible [3].

Integer programming (IP), which is also called integer linear programming (ILP), is a subset of linear programming where decision variables can only get integer values. However, there is no integer restriction on the constants in objective function or constraints. Furthermore, there is a special case of integer programming which

is named 0-1 integer programming or binary integer programming (BIP) where the decision variables need to be either 0 or 1 rather than being any arbitrary integer [4].

IP problems are solved using Simplex method like in LP problems; however, to satisfy the integer requirement, several other advanced algorithms are applied such as cutting-plane method [5], branch and bound [6], branch and cut [7], and branch and price [8].

2.2. Recommendation Systems

Recommendation systems are used in many fields such as electronic commerce, bookstores, movie rental and online streaming companies.

There are two types of mainly used methods in recommendation systems which are content-based and collaborative filtering [9]. In content-based methods, system should have several different profiles and the information of user profiles which are determined based on users' previous preferences. Furthermore, it needs to know which product fits in which profile best. By this way, each user gets a recommendation of products based on their profile. In collaborative filtering (CF), the information of other users' preferences is used to provide a suggestion to the current user. Contrary to content-based systems, the actual content is not used. Instead, the preference information of users are considered [10]. It is first suggested and used by Tapestry mailing system from Xerox Palo Alto Research Center to filter the relevant ones among large number of mails that one receives [11]. Filtering that is based on other users requires a similarity function to find the users which have the similar taste with the current user, and these similarity algorithms differ in each recommendation system due to different needs and properties of systems.

Collaborative filtering is one of the most successful approaches to build recommender systems [12]. It is used to make predictions or give recommendations to a user using the tastes of other users which makes this method collaborative [13]. If a person behaved similarly with some other users in some situations, the assumption is that she

is more likely to have the same opinion with these users again in some other situations rather than with some randomly selected users [10].

There are several different evaluation metrics used to rank different recommendation algorithms such as accuracy over each forecast and utility of a rated list of advised items [10]. Furthermore, there are other aspects to take into consideration when evaluation the performance of a recommendation algorithm such as type, volume, and lifetime of items to be recommended as well as the cost structure of the whole system [13]. Gunawardana *et al.* [9] simplify these aspects and suggest three major groups of different tasks of recommendation systems which are recommending set of interesting items, utility optimization, and prediction of user opinion.

In addition to studies regarding the evaluation metrics of recommender systems, Herlocker *et al.* [14] mention that there are several researchers who believe that these metrics are details, and the user satisfaction should be the bottom-line in recommendation systems. They also claim that, in addition to over-tuning the algorithms to a specific data set, another reason of different recommender systems having almost the same success rate could be a magic barrier that is caused by the inconsistent ratings from users. Backing for this argument, Hill *et al.* [15] say the variance in a user is the barrier of an algorithm which cannot exceed this limitation.

3. PREVIOUS WORK

Integer programming is a widely used method to optimize processes. For example, there are several studies which use integer programming in arranging work shifts in contaminated area with radio-active or chemical materials and football league fixtures in some countries [16,17]. Some publications focus on optimizing the time between final exams in universities [18]. More importantly, many researchers [19–25] have worked on finding an optimized annual academic schedule in educational institutions because it is considered as a difficult problem due to a large number of constraints such as limited number of rooms and day-hour preferences of professors. In another study [26], they try to solve a similar problem for high schools which is smaller in scale but tightly constrained in comparison with the scheduling problem in universities. Moreover, the work [27] concentrates on finding the schedule that leads students to finish their academic program in the shortest time possible. At the same time, they try to allow students to place their courses in specific days of the week and hours of days.

There are also studies proposing different approaches other than integer programming because of the high amount of time and computational power required to solve high dimensional problems. For example, [28,29] try to find the whole calendar of academic institutions using heuristic approaches such as genetic algorithm and particle swarm optimization. Santiago-Mozos *et al.* [30] propose an approach using evolutionary algorithm to assign optimized schedules to all students in an institution. Their approach prioritizes the creation of the schedules of students who come first in their list.

Sediyev, Akboyun, and Uslu [31] develop Soppus which is an online course program planning and recommendation system for Boğaziçi University students. In Soppus, courses are rated by the preference of the user and the system gives optimum schedules based on these ratings and also considering course conflicts. Course programs are suggested using an independent set algorithm which finds a set of n nodes that are not connected in a graph. Nodes, which represent the courses, are connected

if there is a time conflict between them. Proposed course programs are presented in order based on their ranks. The most appropriate course program is given first. In addition, users have the opportunity to view other suggested course schedules.

In this thesis, Mecanin which is an advanced online student course program planning system is proposed. As a further improvement over Soppus, integer programming algorithm is used for course program planning. This allows the system to provide more functionality which in turn makes it more flexible. For example, one additional option in Mecanin is that it gives grouping capability in which students are able to create as many course groups as they want for their course schedule preparation. Also, the minimum and the maximum numbers of courses to be selected from these groups are decided by the user. In Soppus, there were three groups which were designed for the needs of computer engineering students; however, students can personalize their schedule for their grouping needs in Mecanin.

For the recommendation problem, there are many studies about recommendation systems. Some well-known papers focus on movie datasets and use collaborative filtering and regression techniques [32, 33]. Furthermore, Shani *et al.* [34] develop a Markov decision process (MDP) based recommendation algorithm for online bookstores.

Success of these recommendations motivated us to use the idea of these techniques to help students in their elective course selections. Vialardi *et al.* [35] propose a course recommendation method considering demographics, course enrolment, number of courses per semester, average grades, and cumulative grades data of students. They use a statistical classifier algorithm, specifically C4.5 algorithm which is used to generate a decision tree and developed by Ross Quinlan [36], and claim 80% success rate. Ray *et al.* [37] propose a technique for course recommendation based on the previous performance of students. They study 255 students and make item-based and user-based recommendations, and get 30% success rate. Booker [38] designs an online system to suggest courses for prospective students using k-means. Lee *et al.* [39] try to find the association rules between students and courses using previous course preferences of students and course categories. They use curriculum experts to evaluate

their recommendations. Sharon *et al.* [40] design a recommendation system for elective courses using content-based approach considering the requirements of the job market. Bendakir *et al.* [41] develop a course recommendation system for students based on collaborative filtering and machine learning methods considering prerequisites and course evaluations.

In this thesis, we propose a collaborative filtering technique to recommend courses to university students based on the student's and other students' preferences. We, then, adjust the algorithm considering course hour conflicts. We present the results of these approaches. Our results show that similarity based approaches like collaborative filtering are superior to other techniques like popularity based approaches. Our specialized collaborative filtering method and similarity function design give 80% accuracy using specified evaluation metrics. Although there are several recommender system evaluation metrics, we decide to use accuracy since it is the most appropriate metric for offline tests in which historical dataset is being used rather than actual user response. In addition, the only information that the system has is previous user-course selections over 9 semesters. This situation restricted us from using other information such as demographics and department information to give better recommendations. Also, this lack of demographics and department data prevents us from using other proposed course recommendation algorithms which use more data than user-course selection information.

4. MECANIN SYSTEM ARCHITECTURE

Mecanin is a social platform where university students can prepare their course programs and find courses that they might be interested in. There are two main functions of Mecanin: First, it provides a tool which helps students prepare their schedules. For example, the system brings the courses which are matched with the query that can be the code or time of a course, or name of a teacher. Then Mecanin decides a program according to the ratings and conflict choice of the user after adding each course that the user has intended to add to his course program. Second, Mecanin provides course recommendations based on current user's and other users' course preferences. Furthermore, since it is a social platform, a friendship mechanism is provided. Mecanin is implemented in Java, and NEO4J which is a graph based database is used as database management system.

Figure 4.1 represents the Mecanin system architecture. Students and courses are represented as nodes and course selections are represented as edges in graph-based database. Mecanin Server consists of database (DB), database management system (DBMS), and Mecanin Back-End. Mecanin Back-End is implemented in Java and consists of three components which are database access object (DAO), recommendation algorithms, and servlet classes. DAO is implemented for DB operations. Mecanin Client has two parts which are IP Solver that is implemented in JavaScript and Mecanin Front-End.

4.1. Web User Interface

Mecanin has five different types of pages which are register and login, profile, editor, course, and user pages.

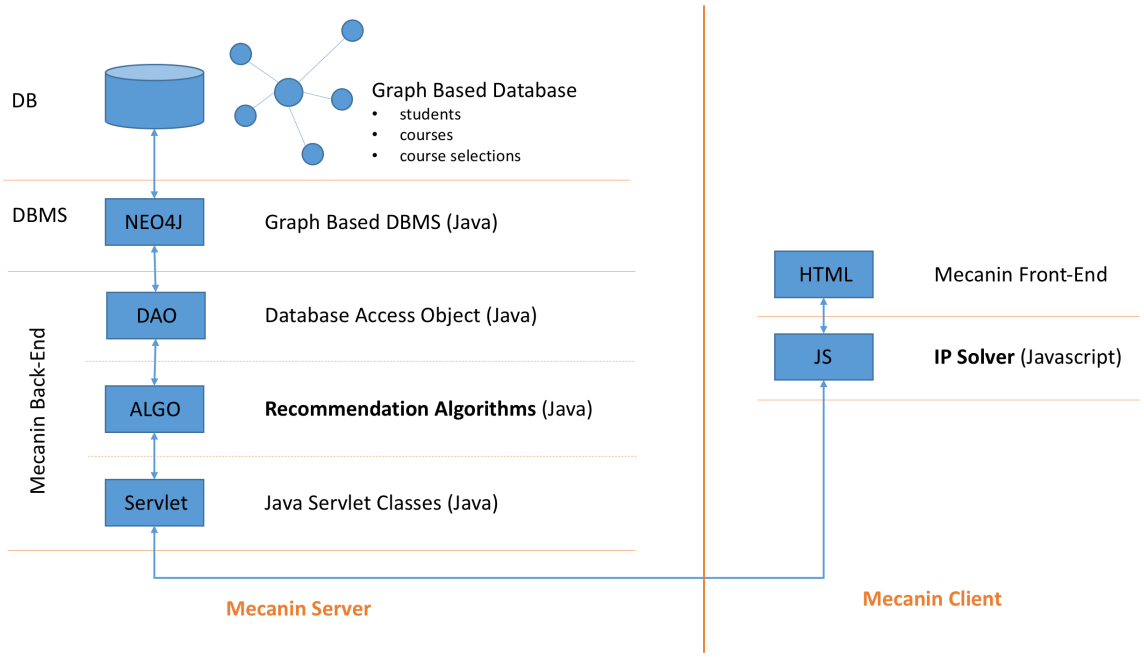


Figure 4.1. Mecanin System Architecture

4.1.1. Registration and Login Page

Since the system requires registration, the first page, as shown in Figure 4.2, is login and register page. Users are required to provide a unique username, a unique e-mail address, and a password. As a register page, users are warned if the username or e-mail address that they provide are not unique. Otherwise, registration completes successfully, and user is logged in right after the registration.

4.1.2. Profile Page

The second page, as shown in Figure 4.3, is called profile page and it appears after logging into the system. In this page, users can see their current course program on a weekly timetable. Moreover, courses, connections to other users, and course recommendations are shown in separate boxes. There is an Edit button above the course list which is used to open the third page, the editor page, for course program planning.

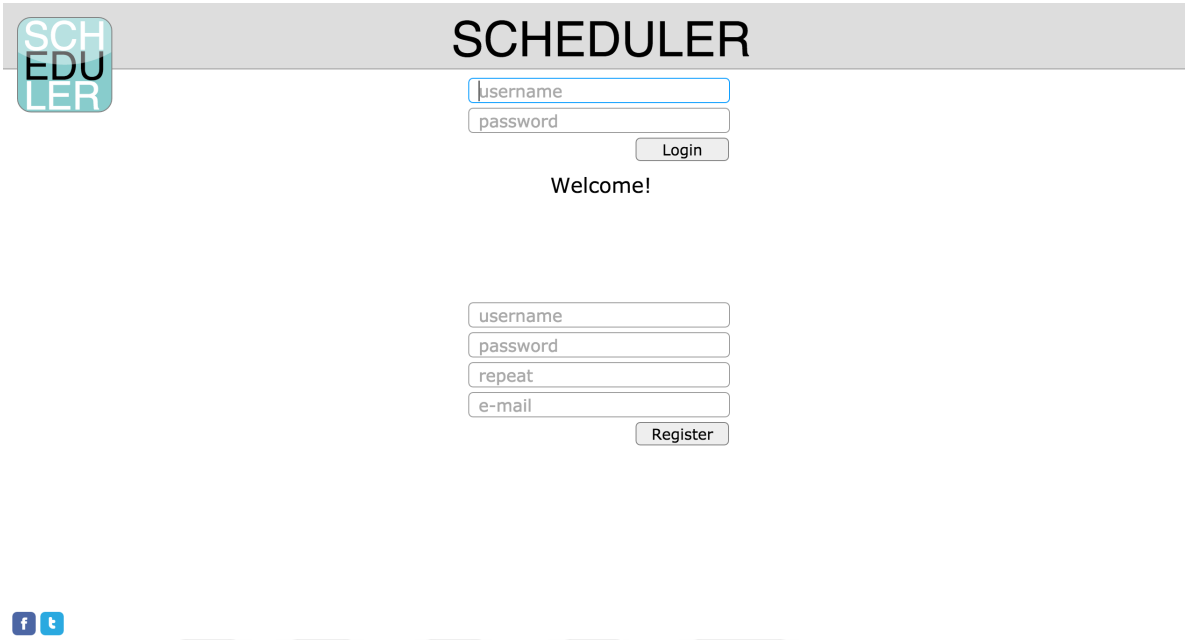


Figure 4.2. Mecanin registration and login page

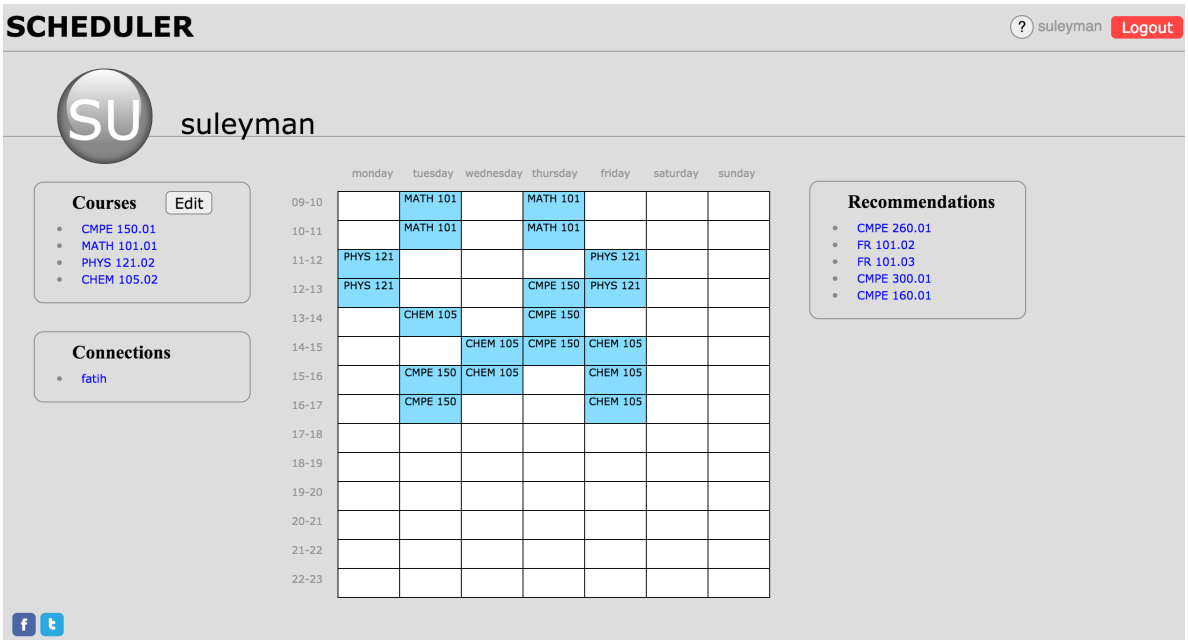


Figure 4.3. Mecanin profile page

4.1.3. Editor Page

Third page, as shown in Figure 4.4, is the editor page where the students prepare their programs. This is the most advanced page of the system. There are two timetables, a search box, and a course list on this page.

First timetable is used to provide a better view of all the courses that the students intend to add to their programs whereas the second one shows the program that Mecanin suggests them. Search box can be used to search courses by their names, abbreviations, teachers and any other property such as date and time. Results which are returned by using the search box appear right below the search box, and users are allowed to select multiple courses at once. After selecting desired courses and clicking the Done button, those courses appear below the search box as an advanced list compared to other lists in the system.

In selected courses list, additional hours of courses such as problem session and laboratory hours are displayed by clicking on courses, and checkboxes are provided for users to select desired additional hours. Furthermore, course ratings are performed in this advanced list. Also, there is a cross symbol which is used to drop currently selected courses and located to the right of the each course in the list.

Mecanin has a star rating system and each course can be rated separately. Courses are rated 2 stars out of 5 by default. Another function of this editor page is that there are two buttons, plus and minus, which are used to add and remove course groups, and each course group has two of those buttons which are used to increase and decrease the minimum and the maximum number of courses to be selected from that group.

4.1.4. Course Pages

Forth page, as shown in Figure 4.5, is the course page. Mecanin has separate pages for each course, and course information such as name and teacher is shown in addition to date and times in this page. Also, course times are displayed in a timetable

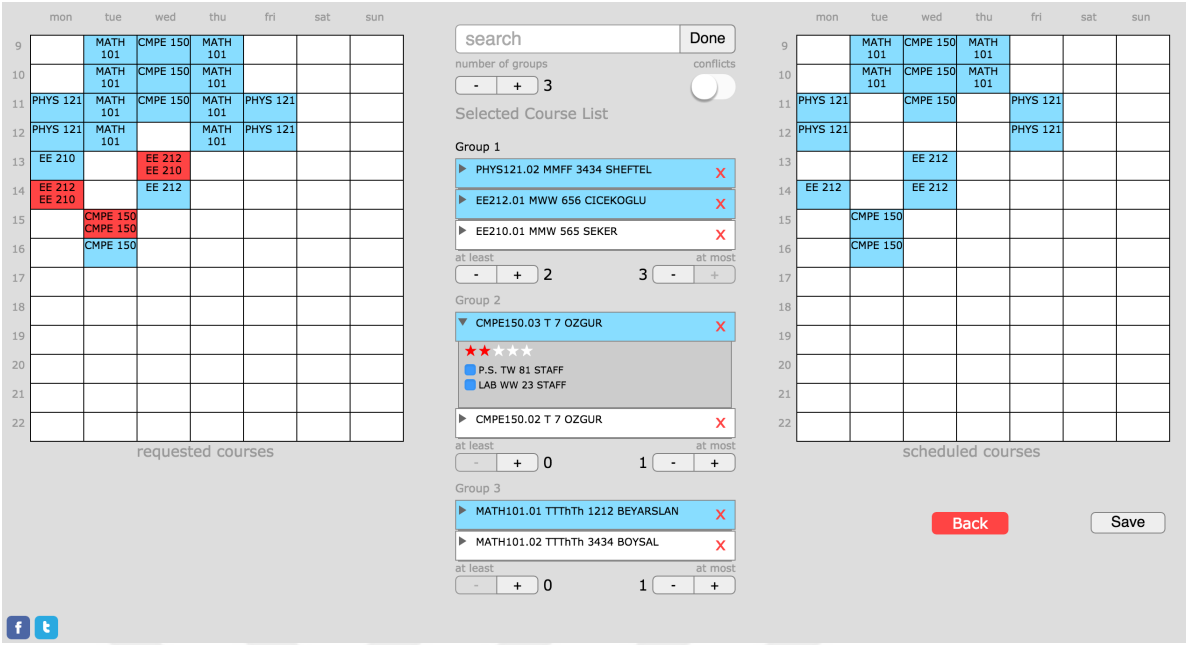


Figure 4.4. Mecanin editor page

with the room information where each class is conducted for a better user experience. Moreover, there is a list of attendance located on the left of the screen which shows the students that are taking this course for the current semester.

4.1.5. User Pages

Fifth page, as shown in Figure 4.6, is called the user page. This page is similar to the first profile page but in this version, other users' profile data is displayed along with courses of the other users as a list and on the timetable as in the profile page. Also, their connections and recommended courses are displayed on this page. In addition, there is a Connect button which is used to connect to other users. Once a user is connected to another user, Connect button changes to Disconnect button which is used for removing the connection to the user. Unlike the profile page, there is no Edit button which is linked to editor page since it is not possible to edit the program of other users.

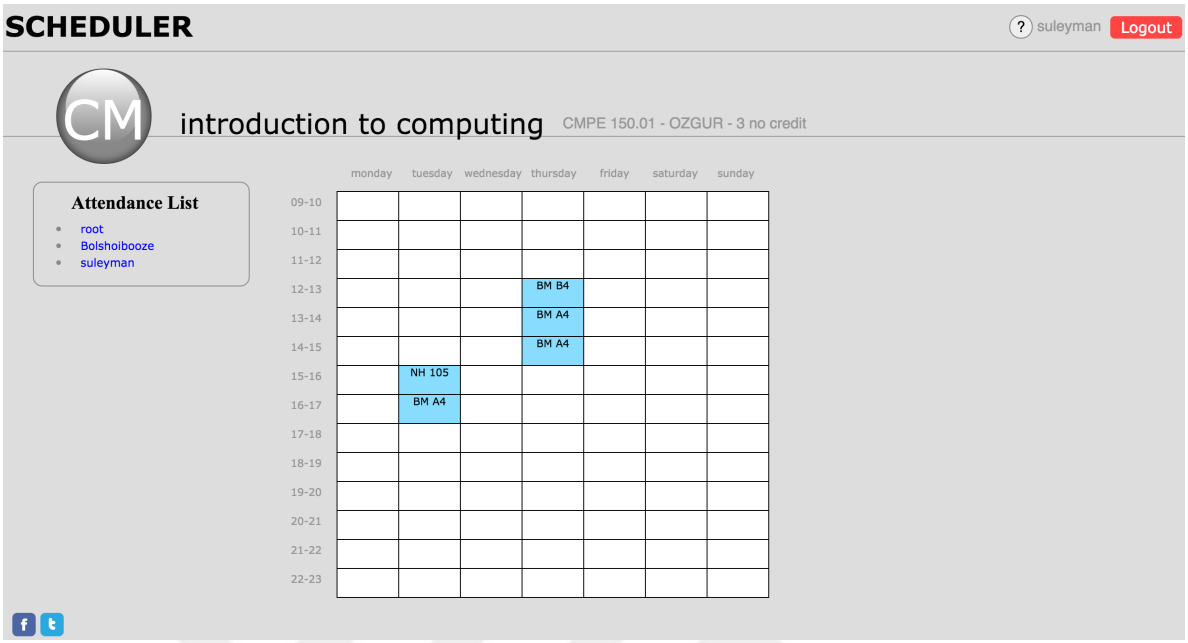


Figure 4.5. Course page design of Mecanin

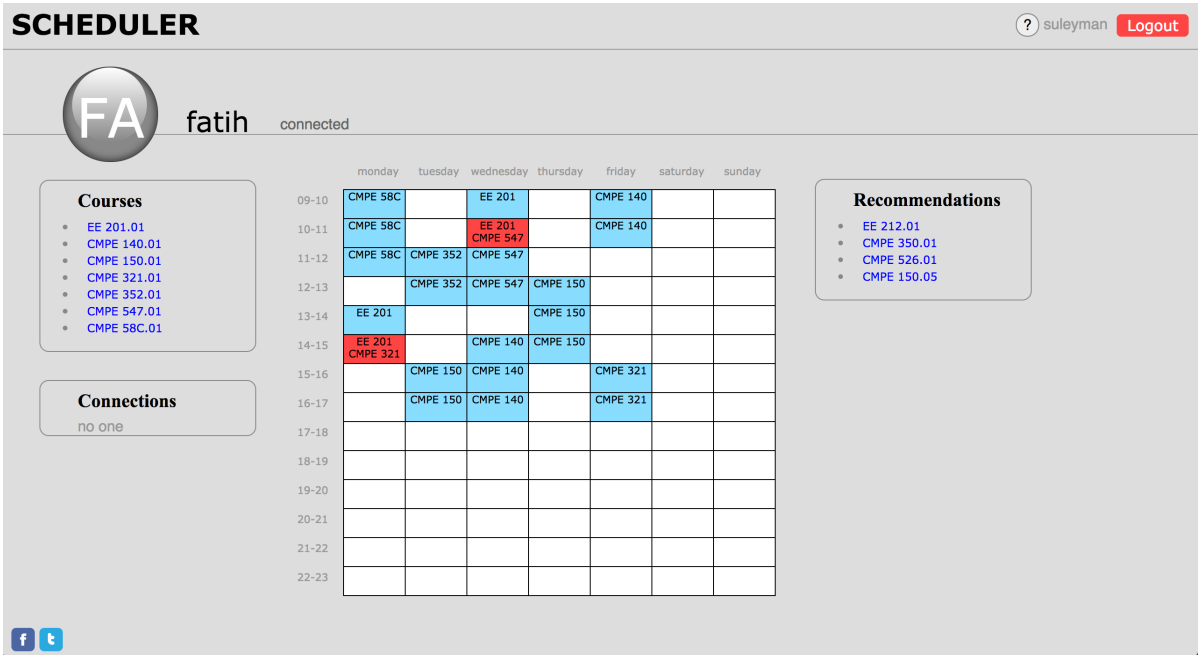


Figure 4.6. Other users' page in Mecanin

4.2. Detailed Usage

In order to use Mecanin system, users are required to register first. They are asked to provide a unique username, a password, and a unique e-mail address. Registered users can login to the system with their usernames and passwords. After logging into Mecanin, profile page appears and current information such as courses, connections, and also recommendations is shown if the system is used and schedule is saved before. The timetable on this page shows the current selected courses. In this page, users can look at the profiles of their connections, visit the pages of their current courses, or check the courses recommended by Mecanin for them. If no course is selected or the user wants to chance his schedule, Edit button can be to open the editor page.

In the editor page, it is necessary to select courses first since all scheduling process is performed here. If the user has already had courses on his schedule, they appear on both timetables and in the course list. Users can start adding courses by typing any information of the intended courses on the search box, and the system finds and brings the courses that are matched with the query that is entered into the search box. Department, code, section, teacher, room, day and hour information could be retrieved from the database. As the users could establish multiple search, they can also select multiple courses from the results of one search. After selecting intended courses, Done button can be clicked to add the selected courses to the timetables and the course list.

Intended courses appear on the left timetable as they do right below the search box as a list after selecting them using the search box. When the conflict option is enabled, system adds all courses on the left timetable to the schedule without doing any integer programming calculations and displays on the right timetable as the suggested course program. By disabling conflicts, Mecanin runs integer programming calculations and suggests the new course program. Courses that are selected for the suggested course program are also shown in blue in the selected courses list which is located in the middle of the screen. If there is a conflict between courses in a time slot, those courses are highlighted in red to draw users' attention to the conflict. If a user wants to remove a course from his list, he can simply click on the cross symbol to the right

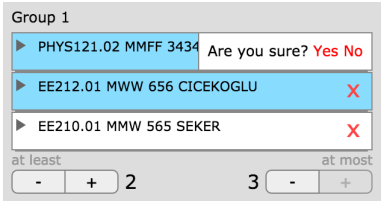


Figure 4.7. Mecanin wants users to confirm their course removals

of the course in selected courses list. For the confirmation purpose, users are asked if they are sure to remove the course from their schedule as shown in Figure 4.7. Once a course is removed, it disappears from both timetables and the selected courses list.

4.2.1. Grouping

Another option that users can benefit from is grouping. By default, there is one group of courses and all selected courses go into this group. Users are allowed to create additional groups and move courses between these groups. Moving process is done by drag-and-drop gestures. The advantage of the grouping mechanism is that it enables users to specify the minimum and the maximum number of courses to be added to the schedule from that group. For instance, a student might be required to take at least one and at most two of the five elective courses determined by his department in a semester in addition to his required courses. He can create a second group that consists of those five courses and specify the minimum and the maximum number of courses as one and two respectively. The system gives program suggestions considering the second group and its limits. The suggested programs include at least one and at most two of the courses from the second group in addition to courses from other groups. Total number of groups cannot exceed the total number of courses because having an empty group does not make any sense. Also, the group that exists from the beginning which is called the default group cannot be removed. Figure 4.8 represents an example of three groups in total.



Figure 4.8. An example of grouping in Mecanin

4.2.2. Ratings

Another criterion that the system considers while providing the optimum schedule to users is the importance of the added courses. As shown in Figure 4.9, Mecanin has a star rating interface for users to decide the importance of each course that they intend to take. This information is supplied by the users and specified as discrete numbers from 1 to 5. As the importance of a course increases, the system is more likely to put that course into weekly schedule. Each course can be assigned importance values, which are 2 by default, separately.

4.2.3. Additional Course Hours

In addition to the scheduling options like grouping and importance, Mecanin is also able to handle optional hours of a course. As shown in Figure 4.10, there can be lab hours or problem sessions assigned to a course which can be optional or mandatory. Users can select these optional hours and add them to their course program. All of these additional hours are selected by default. If the user does not intend to take some

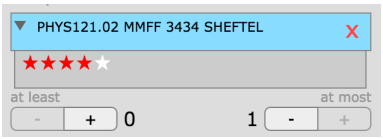


Figure 4.9. Star rating system to specify the importance of a course in Mecanin

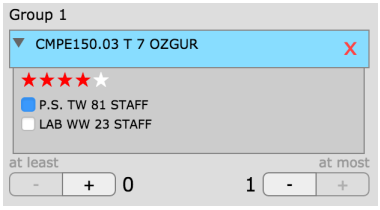


Figure 4.10. Lab and PS hours of a course

or any of these sessions, they need to click on the checkbox and deselect the undesired optional hour.

4.3. Recommendation

Another feature of Mecanin social platform is that it recommends courses to its users. The algorithm used for this purpose is discussed in more detail in Chapter 7. In general, proposed recommendation method is based on the currently selected courses of the user and course selections of other users which have common courses with the current user. Those recommendations are given in the profile page. There is a list to the right of the time table in the profile page. The list consists of five most recommended courses.

4.4. Concepts

Mecanin is designed as an inter-university platform so that it can serve students of other universities in the future. However, universities may have different rules for course plans. Therefore, there are several practices on how Mecanin handles issues that may be faced.

4.4.1. Course Names

Every course having different abbreviation and code number in the university registration system is treated as a standalone course. For instance, CMPE 150 could be renamed as, say, EE 150 or CE 150 for students of a different department having the same time slots, rooms and teachers but they are considered as different courses in Mecanin, so that a user can add both at the same time. However, their conflict will be taken into consideration while suggesting a schedule to the user.

4.4.2. Course Sections

Courses can also have different sections such as CMPE 150.01 and CMPE 150.02. They are also considered as different courses and users are allowed to add them to their programs. Since, for most of the cases, taking only one section of a course with many sections is sufficient, users are advised to create a separate group, put all different sections of a course into that group, and specify the selection limits as at least and at most one.

4.4.3. Additional Hours

Apart from the regular course hours, some courses can have optional or mandatory additional hours such as laboratory work or problem sessions. These additional hours can be selected under the course options pane. In some courses, these additional hours are mandatory. Since the system adds all of these hours by default, it works well for this situation. However, sometimes there maybe more than one additional hours which are complementary to each other. At this point, users need to decide which additional hours they want to add to their schedule.

5. COURSE PROGRAM SCHEDULING ALGORITHMS

Mecanin system can suggest an optimum schedule to its users based on their course preferences and ratings. Integer programming is used to solve this optimization problem. First the Simplex algorithm is run to find integral solutions and feasible region, and then Branch and Bound technique is used to find integer solutions. In this chapter, the IP model is explained, and the test system and cases are described. Results of the Mecanin scheduling algorithm is shown in Chapter 8.

5.1. IP Model

In this chapter, IP model is described in detail by explaining decision variables, constants, objective function, and the constraints of the IP formulation.

Decision variables correspond to all the courses that students select and indicate the courses to be taken during the semester. There are n courses which are denoted as $x_1, .., x_n$, and $x_i = 0$ means that i^{th} course is not in the solution where $x_i = 1$ implies that i^{th} course is in the solution. A course being in the solution means that Mecanin includes that course in the suggested course program.

Constants are denoted as $c_1, .., c_n$ where n refers to the number of courses. These constants are weights that represent how much the student wants to take the corresponding course. In Mecanin Scheduler, users are allowed to rate a course from 1 to 5 using the star rating system.

The objective of the IP model is to find the combination of courses forming a schedule acceptable by the student with the highest score. Total score of a schedule is calculated by adding up the ratings of each course in the schedule, which are determined by the student. As shown in Equation 5.1, the type of the objective function is maximization.

$$maximize \sum_{i=1}^{n} x_i c_i \tag{5.1}$$

There are three types of constraints in this model which are selection, grouping, and model constraints. First of all, all decision variables are binary because each course can only be selected at most once. With this constraint, borders on the value of decision variables are determined. As shown in Equation 5.2, a decision variable can either be 0 or 1.

$$x_i \in \{0, 1\} \tag{5.2}$$

The second type is called grouping constraints which are used by Mecanin to decide the number of courses in the schedule. By default, there is one group; however, students are allowed to create additional groups. As shown in Equation 5.3, this constraint is used for limiting the number of groups such that the total number of groups, which is represented by g, cannot exceed the total number of selected courses, which is represented by n_s, in the model.

$$1 \leq g \leq n_s \tag{5.3}$$

Minimum number of courses to be selected from the group i, say a_i, can be less than or equal to b_i. b_i is the maximum number of courses to be selected from the group i, and it is also less than or equal to t_i, the number of courses in the group. Combining the previous constraint with this one, a rule appears, which is shown in Equation 5.4,

for each group.

$$1 \leq a_i \leq b_i \leq t_i \leq n_s \tag{5.4}$$

The model constraints are used to determine the upper and lower bounds of number of courses to be included in the schedule from a group. As mentioned previously, t_i denotes the number of courses in the group i where the courses in the group can be denoted as $x_m, .., x_{m+t_i-1}$. As shown in Equation 5.5, the total number of courses to be selected from this group has to be greater than or equal to a_i and less than or equal to b_i, where a_i and b_i denote the limitations of the group as explained in the previous constraint, respectively.

$$a_i \leq \sum_{j=0}^{t_i-1} x_{m+j} \leq b_i \tag{5.5}$$

This constraint shows that Mecanin selects at least a_i and at most b_i courses from the i^{th} group to be included in the course program.

Conflict constraints are added to the IP model for handling course hour conflicts. If the conflicts are not allowed by the user, the system does not suggest a course program which has courses that are conflicting. To make all courses in the schedule free from conflicts, the system first finds all conflicting course couples and add a constraint for each of them. These constraints assure that at most one of each course in a couple is selected.

Consider two courses, A and B, have hour conflicts and they are both selected by the user. Their corresponding decision variables are x_A and x_B. Conflict constraint of

courses A and B is presented in Equation 5.6.

$$x_A + x_B \leq 1 \tag{5.6}$$

This constraint shows that the courses A and B which have conflicting hours cannot be selected; therefore, their decision variables cannot be 1 at the same time.

5.2. Example

Consider a computer engineering student in his 3rd year. He needs to take three required courses, namely CMPE 300, CMPE 322, and CMPE 343. In addition to these required courses, he plans to take up to two elective courses of 4th year. He considers three courses, namely CMPE 475, CMPE 422, and CMPE 451. For this case, the student puts the first three required courses into the default course group. Then, he creates another group and puts the other three elective courses into that group. Since the first group is for required courses, the student specifies the number of courses to appear on the program to be at least and at most 3; therefore, he makes sure that all courses in that group will be on his schedule. For the second group, say, since he is unsure about how many elective courses he wants in his schedule, he specifies the boundaries as at least one and at most two.

If the conflicts are granted in the course program, Mecanin suggests the program based on the maximum number of courses to be allowed in the program for each course group. For this case, the three courses in the first group appears on the program. Also, the first two courses of the second group are selected by Mecanin Scheduler because Mecanin tries to put as much courses as it can into the program respecting the boundaries specified by the constraints. Also, since the student does not define any course preference, courses are selected according to the appearance order in the course list. As shown in Figure 5.1, CMPE 300, CMPE 322, and CMPE 343 are selected to

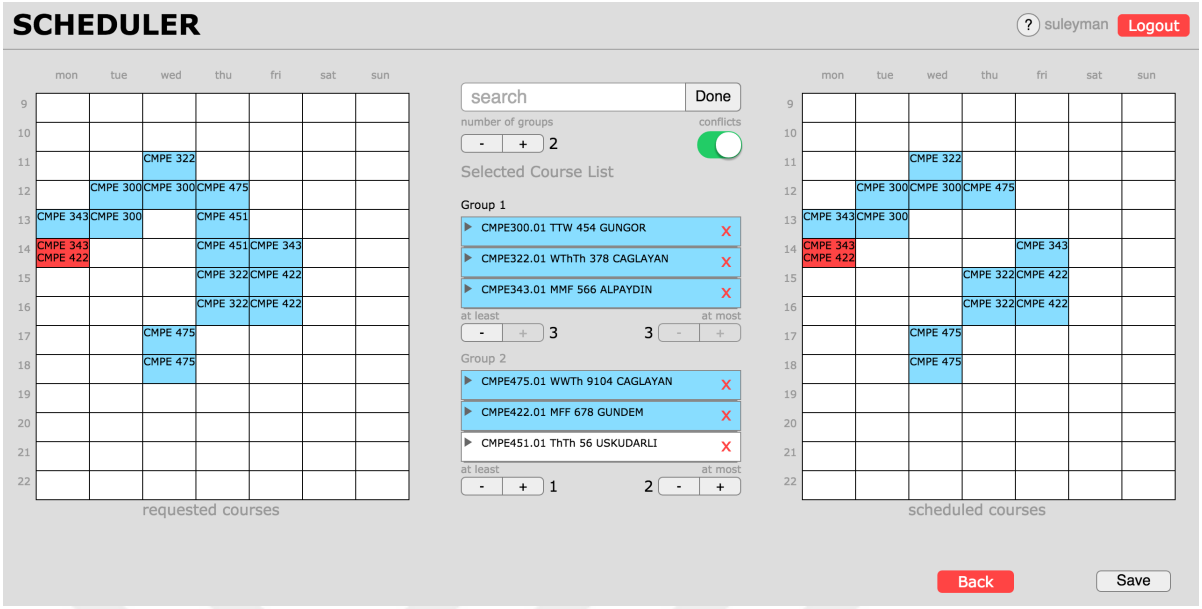


Figure 5.1. Suggested course program by Mecanin Scheduler

be in the recommended schedule together with CMPE 475 and CMPE 422.

Consider the case that the student decides to give the course preference information to the system, say, he has a tendency to take CMPE 422 and CMPE 451 over CMPE 475. In this case, he simply gives higher ratings to the desired courses than the rating of CMPE 475. Since having a high-rated course results in a higher overall score, and the objective of the solver is to maximize the overall score, Mecanin Scheduler suggests the program with higher-rated courses instead of lower-rated ones. As shown in Figure 5.2, CMPE 451 replaces CMPE 475 in the recommended course program.

Consider another case where the student does not want conflicts in his schedule. In both cases, there was a conflict between the courses CMPE 343 and CMPE 422 in the time slot 6 on Monday. When the conflicts are disabled, Mecanin tries to find a schedule without any time conflicts between courses. In this case, as shown in Figure 5.3, CMPE 422 is discarded and CMPE 475 and CMPE 451 find a place in the course program even though CMPE 422 has a higher importance rate than CMPE 475.

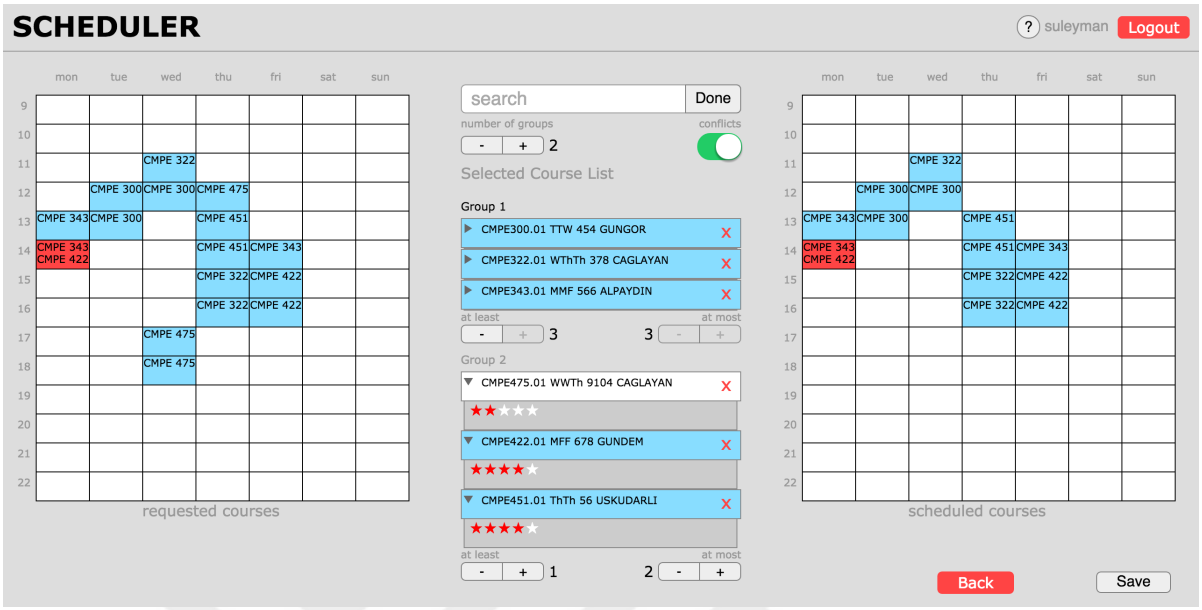


Figure 5.2. Mecanin Scheduler with course preferences

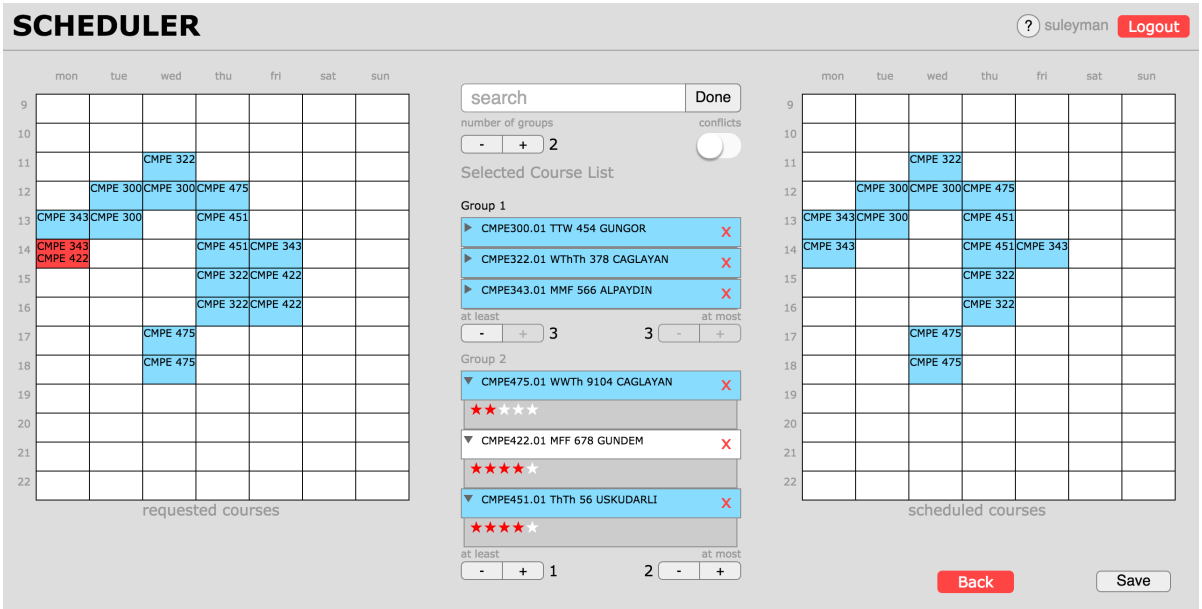


Figure 5.3. Mecanin Scheduler with conflicts disabled

6. GRADUATION PLANNING ALGORITHMS

Besides suggesting a schedule to a student, Mecanin can also help its users prepare their graduation plan. The term graduation plan is used for decision of course programs which leads students to graduate in the minimum number of semesters by taking the remaining courses.

Graduation planning is designed for Department of Computer Engineering (CMPE) at Boğaziçi University but the system can be easily extended to other departments or universities as required information such as curriculum and list of elective courses are supplied.

Algorithm is designed as an IP problem and the details of this IP model is described in Section 6.3.

6.1. Usage

To start using graduation planner, users are asked to enter their completed courses to the system. By skipping this question, which means that the user has not completed any courses yet, the system comes up with a plan for a new student. If the user enters his previously taken courses, Mecanin, first, finds out which courses the user needs to take for his graduation, then it tries to place those courses into semesters, starting from the next semester. Graduation planning algorithm can be used at the beginning of any semester.

As shown in Figure 6.1, completed courses can be selected from the left pane. One semester can be completely selected using the semester checkbox, and elective courses can be selected using the corresponding drop-down menus. There is a summer checkbox on the bottom which is to enable or disable the summer availability of the student. Also, first-order prerequisites of courses are shown next to each course. Moreover, there is an option to determine the maximum number of allowed credits per semester which

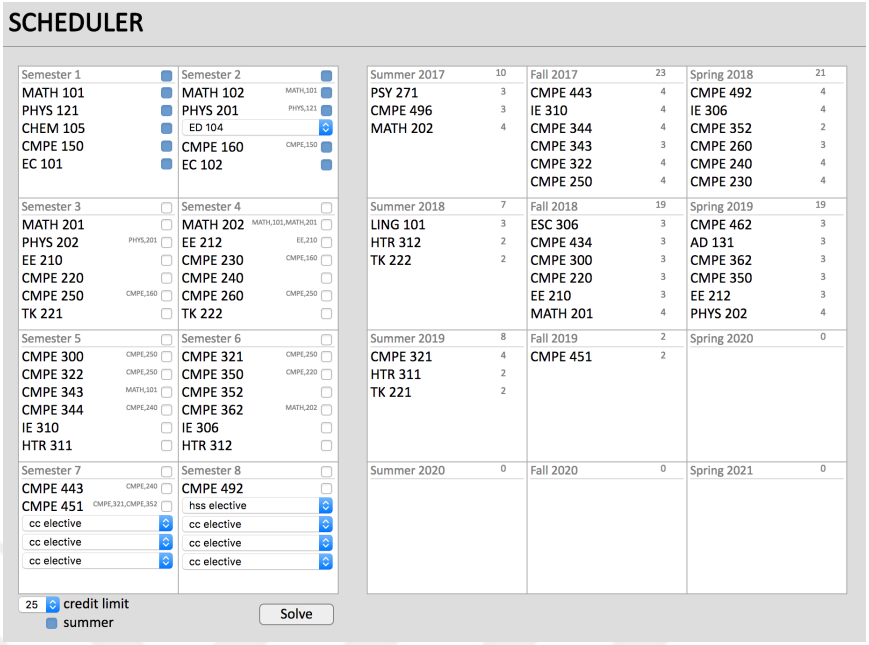


Figure 6.1. Interface of Mecanin graduation planner

applies to spring and fall semesters. After clicking the *Solve* button, recommended graduation plan appears on the right pane. Figure 6.1 also shows an example of a graduation plan of a CMPE student who completed the courses of the first year and allows the planner to include summers into the graduation plan.

6.2. Concepts

There are two types of courses which are required and elective courses. Required courses are determined by the department and each and every one those courses are required to be completed successfully for graduation. For elective courses, students are required to select any courses that they took as an elective course. In the Department of Computer Engineering at Boğaziçi University, there are two types of elective courses. First type is humanities and social sciences (HSS), and the second type is complementary elective (CC). There are two different list of courses for both types of elective course. The requirement for the elective courses is that students need to complete the required number of elective courses for each type. Graduation requirement is that they have to take at least two HSS and six CC courses, three of which must be CMPE courses. However, CC alternatives other than CMPE courses are not taken

into consideration.

Scheduler uses 12 semesters that consists of four summer, spring, and fall semesters which is also equal to four years that is the recommended completion time of the program.

6.3. IP Model

In this chapter, decision variables, objective function, and constraints of the IP model of the graduation planner are presented and explained in detail.

In the IP model of the graduation planner, decision variables are the courses that can be taken by the student. They can either be 0 or 1. 0 means that course is not selected, whereas 1 is used for the selected courses. One of the difficulties of designing integer programming model of this planner is that one variable for each course is not adequate for representing decisions since a decision is not only taking a course but also when to take that course. Therefore, different variables are used for each course in each semester. For example, CMPE 150 is one of the courses that can be taken by a computer engineering student, and $x_{CMPE150}$ is the representation of this course in the system. Since all decision variables have to have a semester information, another number which denotes the semester number is added to this variable. Therefore, $x^1_{CMPE150}$ shows that CMPE 150 is offered in the first semester.

In general, the decision variable of course a of semester s is represented as x^s_a. If there are n different courses in total, there are at most $12n$ decision variables.

Objective of the graduation planner is to find the plan which results in graduation in the minimum number of semesters. Although there is an option for not taking courses in summers, the planner currently does not allow specific semester skipping. It means that it suggests a plan where students take courses in consecutive semesters starting from the upcoming semester. Furthermore, the scheduler fills the semesters with courses up to course and credit limits. Credit limit is 25 for spring and fall

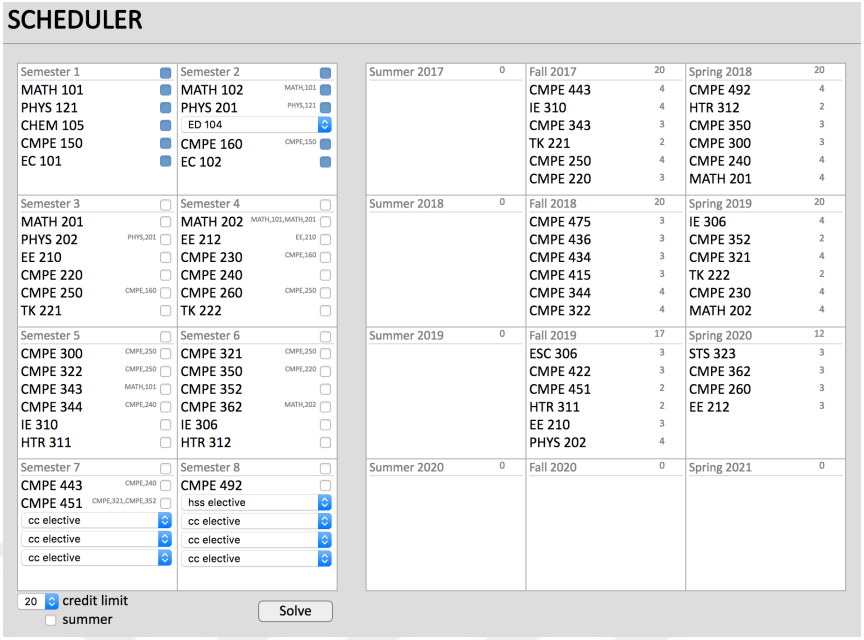


Figure 6.2. Mecanin graduation planner with Summer disabled and 20 credits limit

semesters and 10 for the summer semester by default; however, it can be changed by the user from 20 to 25 for spring and fall semesters. Course limits are 6 for spring and fall semesters, whereas it is 3 for summers. Figure 6.2 shows an example of a graduation plan with summer semesters disabled and semester credit limit selected as 20.

In the objective function, decision variables, which determine whether a course is selected, are multiplied by the coefficient of the semester that they belong to. To ensure the continuity of the semesters, which is basically to forbid skipping one or more semesters, coefficients are given in the decreasing order. Since there are 12 semesters in the planner in total, first semester gets the coefficient 12, second one gets 11 and so on. The last semester's coefficient is 1. As shown in Equation 6.1, the problem becomes a maximization problem.

$$maximize \sum_{a} \sum_{s=1}^{12} x_a^s c_s \ , \ \forall a \in C \tag{6.1}$$

x_a^s represents the decision variable of course a in the semester number s. c_s represents the coefficient of the semester s. C represents the set of all courses.

Required courses are the first type of constraints in the graduation planner. According to the curriculum, there are courses which are required to be taken by the students, and the most important part of the decision process in planning the earliest graduation is assigning these courses, which can be taken up until the graduation, to semesters.

$$\sum_{s=1}^{12} x_r^s = 1 \tag{6.2}$$

In Equation 6.2, x_r represents a required course, and this constraint applies for all required courses. The sum of each decision variable of a required course over 12 semesters must be equal to 1 which means that each required course has to be completed before graduation.

The second type of constraints is for HSS courses. Two HSS courses have to be completed successfully as one of the requirements of the graduation. Therefore, the constraint is designed to cover all courses that are taken during 12 semesters.

$$\sum_{s=1}^{12} \sum_{h} x_h^s \geq 2 \ , \ \forall h \in C_{HSS} \tag{6.3}$$

In Equation 6.3, x_h represents an HSS course, and C_{HSS} represents the set of all HSS courses. All HSS courses that a student would take before graduation must be equal to or greater than 2. Since the planner tries to find the plan with the minimum number of semesters, it will not suggest taking more than two HSS courses.

The third type of constraints is for CC courses. In total, six CC courses have to be completed for graduation. The constraint for CC courses is designed such that students have to take six CC courses over 12 semesters.

$$\sum_{s=1}^{12} \sum_{c} x_c^s \geq 6 \ , \ \forall c \in C_{CC} \tag{6.4}$$

In Equation 6.4, x_c represents a CC course, and C_{CC} represents the set of all CC courses. Total number of CC courses that a student takes over 12 semesters must be equal to or greater than 6. Similar to HSS courses, graduation planner suggests taking exactly six CC courses since it is enough to complete six of them for graduation.

Fourth type of constraints which is called Semester Course Limits is used to limit the number of courses that can be taken during one semester. This is a part of the university regulation to forbid students to overload their course programs. In the proposed graduation planner, this limit is 6 for Fall and Spring semesters and 3 for Summer semesters. Although there are special cases in which students can take more courses, these numbers are used to give a more natural semester planning for students who stick with the recommended course plans.

$$\sum_{i} x_i^s \leq 6 \ , \ \forall s \in \{fall, spring\} \tag{6.5}$$

$$\sum_{i} x_i^s \leq 3 \ , \ \forall s \in \{summer\} \tag{6.6}$$

In Equations 6.5 and 6.6, x_i^s represents the courses that are available for the semester s. In Equation 6.5, the constraint applies for Fall and Spring semesters. In Equation 6.6, the constraint applies for Summer semesters. Furthermore, since the graduation planner tries to calculate the minimum number of semester for graduation, it adds as many courses as it can to the semesters.

Similar to the Semester Course Limits constraint, there is also another constraint which is called Semester Credit Limits and based on the number of total credits of courses in a semester. 25 credits for Fall and Spring semesters and 10 credits for Summer semesters are predefined as the credit limits in the graduation planner. Even though students might be able to take more credits in a semester in some cases, these numbers are used to provide a better plan for regular students.

$$\sum_i x_i^s n_i \leq 25 \ , \ \forall s \in \{fall, spring\} \tag{6.7}$$

$$\sum_i x_i^s n_i \leq 10 \ , \ \forall s \in \{summer\} \tag{6.8}$$

In Equations 6.7 and 6.8, x_i^s represents the courses in the semester s and n_i represents the credits of each course. In Equation 6.7, the constraint applies for Fall and Spring semesters, so that students are now allowed to take more than 25 credits for those semesters. In Equation 6.8, the constraint applies for Summer semesters in which the number of maximum allowed credits is 10.

Likewise course limits, graduation planner tries to fill up the semesters with courses as much as possible to come up with the shortest time to graduate. Therefore, there is not any lower limit on semester credits as a constraint.

Prerequisites are another type of the constraints. According to Boğaziçi University registration system, some courses can only be added to the course program if the prerequisites are satisfied which means that some courses have to be completed. In graduation planner, it is required to take this prerequisites into account such that the planner considers the order of courses that are being taken.

Consider A and B are two courses where A is the prerequisite of B, that is course B can only be added to the course program if A is completed successfully. x_A^i is the decision variable of course A in semester i where x_B^i is the same variable for course B.

Sum of decision variables x_A^j, where $1 \leq j < i$, has to be greater than or equal to x_B^i, for all i where $1 \leq i \leq 12$ which means for all semesters.

$$x_B^i \leq \sum_{j=1}^{i-1} x_A^j \ , \ \forall i \in [1, 12] \tag{6.9}$$

As can be shown in Equation 6.9, for every A-B course pairs, where A is prerequisite of B, sum of decision variables of course A over semesters up to $i-1$ which is $\sum_{j=1}^{i-1} x_A^j$ has to be greater than or equal to the decision variable of course B in semester i which is x_B^i.

Therefore, there are 12 additional inequalities for each course couple with prerequisite relation as for prerequisite constraints.

Conflicts are important part of the graduation planner since students are not allowed to prepare their course program with more than a fixed number of conflicts. In this version of Mecanin graduation planner, conflicts are totally disabled which means that the system finds a plan without any conflicts.

Consider two courses A and B which have conflicting hours in semester i. As shown in Equation 6.10, sum of decision variables of both courses must be less than or equal to 1.

$$x_A^i + x_B^i \leq 1 \tag{6.10}$$

Constraint is designed in such a way that two courses which have at least one hour conflict cannot be selected together. These constraints are added to the integer programming for every semester for each conflicting course couples.

7. COURSE RECOMMENDATION ALGORITHMS

Another feature of the Mecanin system is to provide its users course recommendations which are shown on the profile page. The users can add these courses to their programs using the recommended courses list.

Mecanin recommendation engine has an algorithm based on collaborative filtering. The system recommends courses of users with similar tastes regarding course selections to the current user. In Mecanin, similar taste is measured as the number of common courses. Recommendation engine first finds the courses of the user who will get recommendations. Then, other users who selected at least one of these courses are retrieved, and the courses of those users, other than the ones that the current user already has, are selected from the database. This course list becomes the base list for recommendation. Since NEO4J, the graph based database, is used, this operation does not require table joins as in the conventional relational databases.

Consider G is the main graph which consists of all courses and students in a semester and represents student-course selections, and v is the node which represents the current student for which the system is suggesting courses. Pseudo code of the algorithm of Mecanin recommendation engine is presented in Figure 7.1 where w is the courses of v, e is the other users who select w, x is the courses of e, and s_{ve} is the number of common courses between v and e.

Mecanin recommendation engine has an internal ordering algorithm to order courses that are going to be recommended. For instance, as shown in Figure 7.2, suppose a current user, U1, who is to receive recommendations, has currently five courses, namely C1, C2, C3, C4, and C5, and there are three other users U2, U3 and U4. Suppose, U2 has C1 and C2 as common courses with U1 and has taken C6 additionally. Moreover, suppose that U3 has C3 as a common course with U1 and C7, C8, and C9 as other courses in his program. Lastly, U4 has C4 and C5 as common courses with U1 and C7 and C9 additionally. First of all, the system will select courses C6,

Function Recommend Courses

Inputs: Graph G, node v
Goal: Form the list of recommended courses
Output: $L_{recommendations}$, weighted list of courses
start from the current node v
for each node w in selects list of node v **do**
  for each node e in selected by list of node w **do**
    Set s_{ve} common course count of v and e
    for each node x in selects list of node e **do**
      if x is not selected by v **then**
        if x is NOT in $L_{recommendations}$ **then**
          add x to $L_{recommendations}$ wrt. weight s_{ve}
        else
          update x in $L_{recommendations}$ wrt. weight s_{ve}
        end if
      end if
    end for
  end for
end for
return $L_{recommendations}$

Figure 7.1. Mecanin Course Recommendation Algorithm

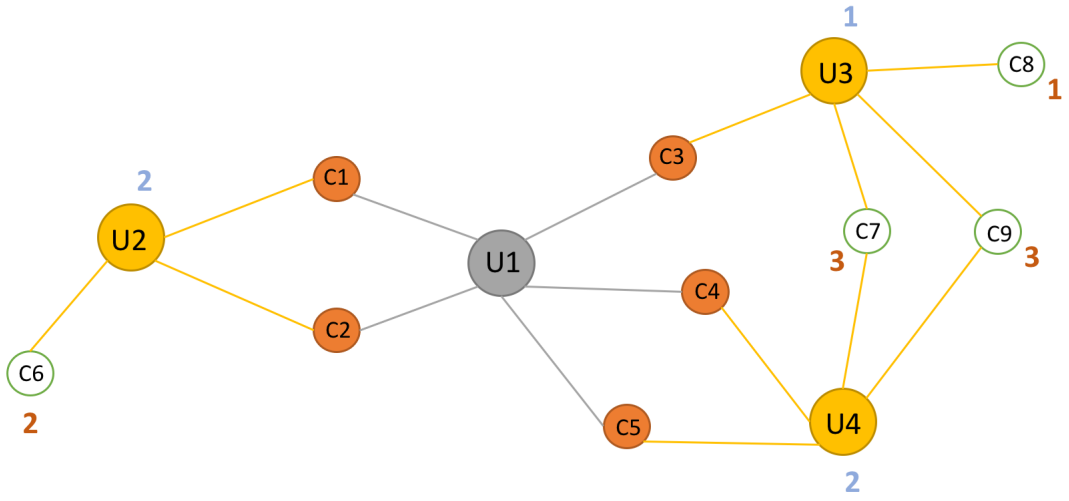


Figure 7.2. Graph representation of course selections

C7, C8, and C9 as the base recommendation list. Although the size of this list is 4 for this example, it becomes larger in real world cases since there would be considerably many students and courses. To decide which courses to recommend from the base list, recommendation engine first assigns U2, U3, and U4 scores based on the total number of common courses with the current user.

Figure 7.2 shows the graph formed as a result of the aforementioned example. In this example, U2 and U4 have two courses and U3 has one common course with U1. After calculating scores for users, system calculates the scores for the courses in the base list. The score for C6 becomes 2 because it is only selected by U2 which has a score of 2. C7 and C9 would have a score of 3 because they are both selected by U3 and U4 which have scores of 1 and 2 respectively. Finally, the recommendation score for C8 would be 1 because of U3.

Our first recommendation algorithm, which is denoted as CF, sorts the courses in the base list based on their scores and recommends the first 5 or 10 courses according to the list size preference. A new variant of this algorithm, which is denoted as CF-0, takes time conflicts into account, eliminates those conflicting courses, and does not include them in the recommended courses list. A conflict between two courses means

that a student cannot attend all hours of both courses, yet he needs to make a choice between them. A third variant, which is denoted as CF-1, is similar to the second algorithm but it allows courses which have at most one hour conflict with the current courses of the user to be recommended.

There are three other algorithms which are used for comparison reasons. The first algorithm is random course recommendation algorithm (RND). In the second algorithm which is called roulette wheel algorithm (RWH), courses are placed in a roulette wheel according to their number of selections by the users. Then, recommendations are made according to the numbers that are in the roulette wheel. The third algorithm is a popularity based algorithm (POP) which sorts the courses according to their number of selections and recommends first n courses. CF is the novel algorithm developed based on collaborative filtering as described in Chapter 7. CF-n, where n denotes the maximum number of total allowed conflicts in the program, is the modified version of CF where the courses that cause conflicts are eliminated. In this thesis, only CF-0 and CF-1 variants are considered.

8. TESTS AND RESULTS

8.1. Mecanin Scheduler

Several test cases are prepared to measure the effectiveness of the Mecanin Scheduler. These cases differ in (i) the number of courses, (ii) whether they allow time conflicts between courses, (iii) course number constraints on groups as well as the number of groups. Cases are named according to these differences. Following is the format of case naming:

$$case < course\ number >< conflict >< group\ number >$$

For example, the name of the case with 15 courses, no conflict with two groups is *case15n2* and 20 courses with conflicts allowed with one group is *case20a1*.

Mecanin Scheduler is tested against LPSolve which is implemented in ANSI C. The test system used is Early-2013 Macbook Pro with Intel Core i7 3.0 GHz processor (3540M) and 8GB memory. Only the times for solving the IP problem is considered. Reading input files, parsing them or generating required classes are not included in the compared times.

To measure the speed of the solvers, all cases are run on both solvers. Results that are shown in the table are the run time results in milliseconds of running the each case 1000 times.

As shown in Table 8.1, Mecanin Scheduler runs faster than LPSolve up to 20 courses, but when the number of courses increases, LPSolve starts solving the linear problems faster. However, in the real life, a reasonable estimation could be that an average student probably does not attempt to add more than 20 desired courses to generate a course program. Furthermore, Mecanin Scheduler works faster when the

Table 8.1. Solution times in milliseconds

Cases	**Mecanin**	**LPSolve**
case5a1	26	346
case5a2	27	257
case5n1	27	325
case5n2	27	334
case10a1	54	367
case10a2	70	333
case10n1	87	340
case10n2	81	378
case15a1	111	219
case15a2	82	261
case15n1	151	377
case15n2	119	283
case20a1	219	363
case20a2	122	379
case20n1	322	371
case20n2	197	300
case25a1	339	395
case25a2	189	454
case25n1	493	496
case25n2	321	355

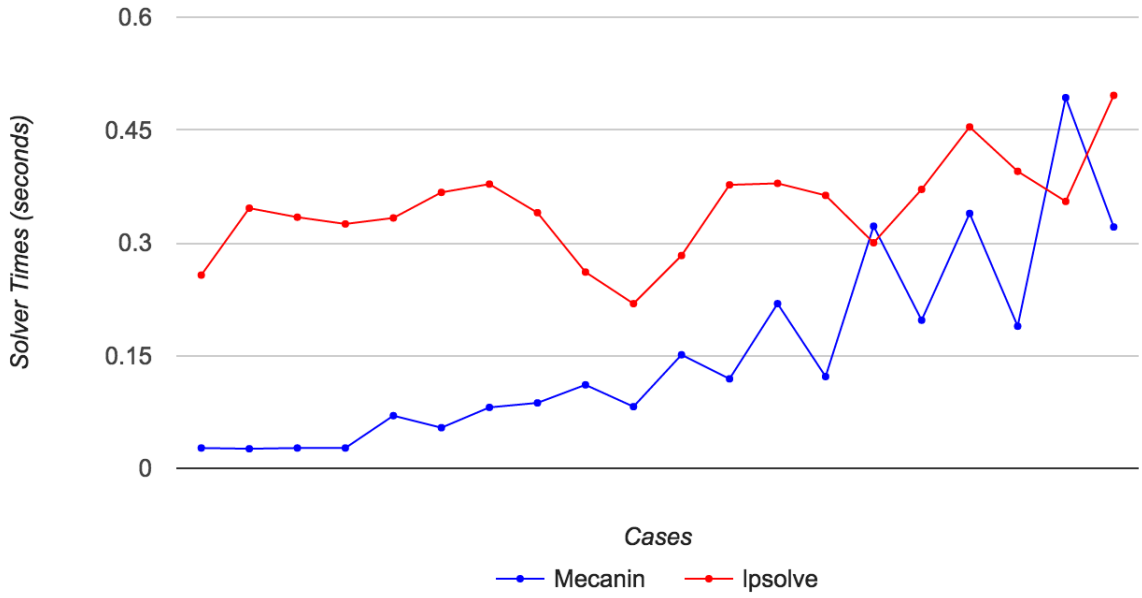


Figure 8.1. Comparison of solving times

courses are divided into groups. In other words, it works slower when all desired courses are put into one group where user does not create additional groups to inform the system further. Comparison of the two implementations is shown in in Figure 8.1.

Additionally, Mecanin is designed as a web based social platform and Mecanin Scheduler is implemented in JavaScript whereas LPSolve is implemented in ANSI C. Therefore, Mecanin Scheduler could be faster compared to the current results when it is implemented in C. However, the reason behind implementing scheduler in JavaScript is to shift the server load to clients.

8.2. Recommendation Engine

Mecanin recommendation engine is tested with the data that is collected from Boğaziçi University students. Students' course statistics are gathered from the Soppus platform, which is the first attempt to build an online course-based social platform for university students [31]. Test is conducted on the data between the years 2010 and 2014. Due to the small of number of courses, users, and courses per user, summer semesters are disregarded and are not considered during the test.

Table 8.2. Student and course statistics over 9 semesters

Semester Statistics	**min.**	**avg.**	**max.**
No. of active courses	776	1009	1203
No. of users	438	898	1448
No. of users with more than one course	423	484	1428
No. of courses selected by users	5.9	6.3	6.4
Total no. of course selections	2576	5639	9291

8.3. Statistics

In total, 9 semesters, which are either spring or fall, are used for testing the recommendation engine. No summer semester data is used. The first used semester is Spring 2010 and the last one is Spring 2014. Table 8.2 shows various statistics of the data that is used. Note that the total number of course selections over 9 semesters is over 50K which is also the total number of course recommendations that is made for each algorithm.

8.4. Test Methods

As described in the recommendation algorithms section, main and modified versions of Mecanin recommendation engine are tested. In addition, results of random course recommendation and recommendation based on popularity of courses are also provided. The test method is based on removing an already selected course of a user and trying to estimate the course based on the remaining courses of the user and the other users who have common courses with the current user. Success of a recommendation is measured in such a way that if the dropped course appears in the suggested courses list, corresponding recommendation is counted as successful. In total, the success of the algorithm is calculated as the fraction of course selections that is successfully estimated. Tests are performed using variable size of recommendation lists.

Table 8.3. Comparison of recommendation algorithms (ratio of successful course recommendations)

	Size of the recommended courses list					
Methods	**1**	**2**	**3**	**4**	**5**	**10**
RND	-	-	-	-	0.031	0.062
POP	-	-	-	-	0.260	0.350
RWH	-	-	-	-	0.076	0.140
CF	0.610	0.706	0.748	0.783	0.807	0.874
CF-0	0.613	0.706	0.750	0.784	0.808	0.875
CF-1	0.612	0.707	0.749	0.810	0.810	0.876

8.5. Test Results

Results are shown in Table 8.3. First, the scores for the non-similarity based algorithms are given. RND, POP, and RWH algorithms are tested using the recommendation list sizes 5 and 10, where as CF, CF-0, and CF-1 are tested using the list sizes 1 to 4 in addition to 5 and 10.

9. CONCLUSIONS

In this thesis, two integer programming models are proposed for university students. First one is to optimize their course programs based on their course ratings and course hour conflicts. The second one is to prepare their graduation plan which shows the distribution of their remaining courses over semesters. Furthermore, an online web platform, Mecanin, is built to provide a web interface of the tools that are optimum schedule tool and graduation planner to Boğaziçi University students. It is observed that these tools which either are implemented in JavaScript or use LPSolve as IP solver are sufficiently responsive to suggest a schedule for students based on their course ratings and time conflicts of courses, and also provide a plan for graduation.

Moreover, a recommendation algorithm which is based on collaborative filtering method is proposed. This algorithm considers the current courses of the user and the course preferences of other users which may be similar to the user in term of course programs. Several variants of this algorithm is also provided, and the results of these proposed algorithms and some other methods which are not based on similarity techniques are compared. It is observed that CF, which is a similarity based method, outperforms all other methods. In addition, it is shown that by considering the conflicts, the score of the CF method could be improved.

9.1. Future Work

As for future work, there can be several visual and usability improvements and additional functionalities to be added to Mecanin. One of the visual and usability improvements is about removing course groups. Currently, course group are removed by decreasing the number of the groups which results in erasing the last group and transferring the courses in the last group to the group which is the second to the last. This can be improved such that each group can be removed individually and if the group is not empty, there can be an additional confirmation for either removing the courses in that group or transferring those courses to another group.

Since the recommendation algorithm of Mecanin uses limited data for course recommendation, another improvement could be adding more parameters to increase the success rate of course recommendations. Mecanin already has a connection capability, a variant of friendship mechanism, that can be added to the recommendation algorithm. Therefore, users would get some of the course recommendations based on their friends on the system.

In addition, there are several other possible improvements regarding the graduation planner. First, current version of Mecanin tries to find a plan for the earliest graduation taking the semester course and credit limits into consideration. Although it gives the option to select the credit limit for spring and fall semesters between 20 and 25, students might still be interested in manipulating the maximum number of allowed credits for each semester separately. Also, Mecanin has the option of skipping summer semesters for students who do not like to take courses in summers. One improvement for semester skipping would be to have the option to skip each semester individually. In other words, students will be able to select their active semesters, when they want to take courses, individually which would be a useful tool for students who are planning to suspend their enrolment or go to another university for exchange purposes for one or more semesters. One last improvement would be to distribute the courses equally among semesters. Since Mecanin tries to find the schedule which leads to the earliest graduation, sometimes it places just one course to the last semester whereas the other semesters are full of courses. A student might desire to have semesters which have courses equally distributed among them.

REFERENCES

1. Bazaraa, M. S., J. J. Jarvis and H. D. Sherali, *Linear programming and network flows*, John Wiley & Sons, 2011.

2. Dantzig, G. B., "Application of the simplex method to a transportation problem", *Activity analysis of production and allocation*, Vol. 13, pp. 359–373, 1951.

3. Ferguson, T. S., "Linear programming: A concise introduction", 2000, `http://www.math.ucla.edu/~tom/LP.pdf`, accessed at July 2017.

4. Garfinkel, R. S., G. L. Nemhauser *et al.*, *Integer programming*, Vol. 4, Wiley New York, 1972.

5. Kelley, J. E., Jr, "The cutting-plane method for solving convex programs", *Journal of the society for Industrial and Applied Mathematics*, Vol. 8, No. 4, pp. 703–712, 1960.

6. Land, A. H. and A. G. Doig, "An automatic method of solving discrete programming problems", *Econometrica: Journal of the Econometric Society*, pp. 497–520, 1960.

7. Padberg, M. and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems", *SIAM review*, Vol. 33, No. 1, pp. 60–100, 1991.

8. Savelsbergh, M., "A branch-and-price algorithm for the generalized assignment problem", *Operations research*, Vol. 45, No. 6, pp. 831–841, 1997.

9. Gunawardana, A. and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks", *Journal of Machine Learning Research*, Vol. 10, No. Dec, pp. 2935–2962, 2009.

10. Breese, J. S., D. Heckerman and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering", *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52, Morgan Kaufmann Publishers Inc., 1998.

11. Goldberg, D., D. Nichols, B. M. Oki and D. Terry, "Using collaborative filtering to weave an information tapestry", *Communications of the ACM*, Vol. 35, No. 12, pp. 61–70, 1992.

12. Ricci, F., L. Rokach and B. Shapira, "Introduction to recommender systems handbook", *Recommender systems handbook*, pp. 1–35, Springer, 2011.

13. Resnick, P. and H. R. Varian, "Recommender systems", *Communications of the ACM*, Vol. 40, No. 3, pp. 56–58, 1997.

14. Herlocker, J. L., J. A. Konstan, L. G. Terveen and J. T. Riedl, "Evaluating collaborative filtering recommender systems", *ACM Transactions on Information Systems (TOIS)*, Vol. 22, No. 1, pp. 5–53, 2004.

15. Hill, W., L. Stead, M. Rosenstein and G. Furnas, "Recommending and evaluating choices in a virtual community of use", *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 194–201, ACM Press/Addison-Wesley Publishing Co., 1995.

16. Sawik, T., "An integer programming approach to scheduling in a contaminated area", *Omega*, Vol. 38, No. 3, pp. 179–191, 2010.

17. Recalde, D., R. Torres and P. Vaca, "Scheduling the professional Ecuadorian football league by integer programming", *Computers & Operations Research*, Vol. 40, No. 10, pp. 2478–2484, 2013.

18. MirHassani, S., "Improving paper spread in examination timetables using integer programming", *Applied Mathematics and Computation*, Vol. 179, No. 2, pp. 702–

706, 2006.

19. Fiallos, J., "Integer Programming Model for Course Scheduling at UNITEC", *IIE Annual Conference. Proceedings*, p. 2048, Institute of Industrial Engineers-Publisher, 2009.

20. Daskalaki, S. and T. Birbas, "Efficient solutions for a university timetabling problem through integer programming", *European Journal of Operational Research*, Vol. 160, No. 1, pp. 106–120, 2005.

21. MirHassani, S., "A computational approach to enhancing course timetabling with integer programming", *Applied Mathematics and Computation*, Vol. 175, No. 1, pp. 814–822, 2006.

22. Dimopoulou, M. and P. Miliotis, "Implementation of a university course and examination timetabling system", *European Journal of Operational Research*, Vol. 130, No. 1, pp. 202–213, 2001.

23. Daskalaki, S., T. Birbas and E. Housos, "An integer programming formulation for a case study in university timetabling", *European Journal of Operational Research*, Vol. 153, No. 1, pp. 117–135, 2004.

24. Acosta-Amado, R. J. and M. Villa-Marulanda, "An Integer Programming Model for The Academic Timetabling Problem", *IIE Annual Conference. Proceedings*, p. 2712, Institute of Industrial Engineers-Publisher, 2013.

25. Bakir, M. A. and C. Aksop, "A 0-1 integer programming approach to a university timetabling problem", *Hacettepe Journal of Mathematics and Statistics*, Vol. 37, No. 1, 2008.

26. Boland, N., B. D. Hughes, L. T. Merlot and P. J. Stuckey, "New integer linear programming approaches for course timetabling", *Computers & Operations Research*, Vol. 35, No. 7, pp. 2209–2233, 2008.

27. Patel, M. H., H. J. Tsao and Z. J. Abdollahi, "Integer Programming for Student Program Design and Scheduling", *IIE Annual Conference. Proceedings*, p. 1, Institute of Industrial Engineers-Publisher, 2011.

28. Wang, Y.-Z., "Using genetic algorithm methods to solve course scheduling problems", *Expert Systems with Applications*, Vol. 25, No. 1, pp. 39–50, 2003.

29. Shiau, D.-F., "A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences", *Expert Systems with Applications*, Vol. 38, No. 1, pp. 235–248, 2011.

30. Santiago-Mozos, R., S. Salcedo-Sanz, M. DePrado-Cumplido and C. Bousoño-Calzón, "A two-phase heuristic evolutionary algorithm for personalizing course timetables: a case study in a Spanish university", *Computers & operations research*, Vol. 32, No. 7, pp. 1761–1776, 2005.

31. Sadiyev, E., M. Akboyun and S. Uslu, "Course Schedule Planner", Senior project, Dept. of Computer Engineering, Boğaziçi University, 2010.

32. Ansari, A., S. Essegaier and R. Kohli, "Internet recommendation systems", *Journal of Marketing research*, Vol. 37, No. 3, pp. 363–375, 2000.

33. Mild, A. and M. Natter, "Collaborative filtering or regression models for Internet recommendation systems?", *Journal of Targeting, Measurement and Analysis for marketing*, Vol. 10, No. 4, pp. 304–313, 2002.

34. Shani, G., R. I. Brafman and D. Heckerman, "An MDP-based recommender system", *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pp. 453–460, Morgan Kaufmann Publishers Inc., 2002.

35. Vialardi, C., J. Bravo Agapito, L. S. Shafti and A. Ortigosa, *Recommendation in higher education using data mining techniques*, Barnes, T., Desmarais, M., Romero, C., & Ventura, S., 2009.

36. Quinlan, J. R., *C4. 5: programs for machine learning*, Elsevier, 2014.

37. Ray, S. and A. Sharma, "A collaborative filtering based approach for recommending elective courses", *Information Intelligence, Systems, Technology and Management*, pp. 330–339, Springer, 2011.

38. Booker, Q. E., "A student program recommendation system prototype", *Issues in Information Systems*, Vol. 10, No. 2, pp. 544–551, 2009.

39. Lee, Y. and J. Cho, "An Intelligent Course Recommendation System", *SmartCR*, Vol. 1, No. 1, pp. 69–84, 2011.

40. Sharon, Y.-J. L. and K.-W. Lien, "CareerMaker: An Adaptive Elective Courses Recommendation System", *e-CASE & e-Tech International Conference*, pp. 1118–1125, 2013.

41. Bendakir, N. and E. Aïmeur, "Using association rules for course recommendation", *Proceedings of the AAAI Workshop on Educational Data Mining*, Vol. 3, 2006.