Summer 2022

Personalizing student graduation paths using expressed student interests

Nicolas Charles Dobbins

Department:

PERSONALIZING STUDENT GRADUATION PATHS USING EXPRESSED STUDENT INTERESTS

by

NICOLAS CHARLES DOBBINS

A THESIS

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

2022

Approved by:

Ali R. Hurson, Advisor
Sahra Sedigh Sarvestani, Co-Advisor
Joe Stanley

ABSTRACT

This work proposes an intelligent recommender approach to facilitate personalized education and help students in planning their path to graduation. The original research contribution of this work is to develop a recommender approach that pervasively personalizes and optimizes a student's path to graduation by accounting for the student's career interests and academic background. The approach is a multi-objective optimization problem, subject to institutional constraints, with the goal of optimizing the graduation path with respect to one or more criteria, such as time-to-graduation, credit hours taken, and alignment with student's career interests. The efficacy of the approach is illustrated and verified through its application to undergraduate curriculum in Computer Science, Computer Engineering, and Electrical Engineering at Missouri University of Science and Technology.

The proposed approach differs from others in that it combines personalized, content-based course recommendation and graduation path optimization into one multi-objective optimization problem while also maintaining a design that can easily be applied to various disciplines. The proposed recommender approach is developed as a part of the Pervasive Cyberinfrastructure for Personalized eLearning and Instructional Support (PERCEPOLIS), which is a system designed to offer several tools for assisting both students and advisors in the advising process. With this approach, PERCEPOLIS is able to generate a full path to graduation for a student satisfying both degree and institutional requirements, while reducing time-to-degree.

ACKNOWLEDGMENTS

I would like to express the sincerest of appreciation for the support group that surrounded me throughout this journey, for without them this work would not have been possible. First, I would like to thank my family for their consistent support and love, especially during the strenuous times. Second, I would like to thank my advisors, teachers, and peers, who have provided me with countless learning opportunities and have always been available for questions and collaboration. Finally, I would like to thank my wife, who has not only provided an incredible amount of support, but also stood patient as I finish this collegiate journey.

I would once again like to thank my advisors, Dr. Sedigh and Dr. Hurson, for the opportunity to be not just a scholar of this outstanding research group, but a leader as well. These opportunities have allowed me to grow and progress my skills in the field in a way that I never could have imagined. Thank you.

This research opportunity has increased my appreciation for experimentation, and this opportunity could not have happened without all of you. I am tremendously grateful for what you all have done for me.

This work in part has been supported in part by the National Science Foundation under S-STEM Grant #1742523.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

Education has long been a highly regarded aspect of society, so much so that nearly everyone spends most of the first eighteen years of their life in education. In America, it is standard to have at least a high school diploma, and many jobs are unattainable without it. Technology has begun to make its way into the classroom over the past few decades and has allowed teachers to offer a range of different teaching techniques that cater to students varying needs. As of late, this technology is turning towards creating a personalized learning experience for students at all levels. The National Academy of Engineers has even declared advancing personalized learning as one of its fourteen grand challenges for engineering for the 21st century [1]. Personalized learning has been implemented in several different ways, most often in online learning systems and workplace learning management systems [2] [3]. While a number of approaches have been proposed for the collegiate space, few have made it into universities on a large scale. Further, advising, especially at the college level, is one of the few areas in education where personalized learning is lacking. At many universities, students still have to manually plan out their course schedule, meaning that they typically only are looking one semester ahead in their planning. Very few students begin their college career with knowledge of which courses they will take in each semester, simply because drawing out a full path to graduation is a tedious and time consuming task. Even those that do take the time to plan out a full path to graduation are not able to adequately determine an efficient path that reduces time-to-degree simply due to the number of possible paths. As a result, many students go through their degree just selecting courses according to a fixed and predefined schedule, which can result in course availabilities and scheduling conflicts causing delays in when a course is taken, eventually resulting in a delayed graduation. This method can also lead to students unknowingly selecting course loads far heavier than their expectations and their potential, leading to frustration, a drop in performance, or a delay in graduation date. The aforementioned factors affect overall students' satisfaction with their degree and result in reduced graduation rates and retention rates. In 2020, the retention rate for full-time students at four year universities was 81.6% [4]. This means that one in five students dropout every year, likely due to either financial issues, poor performance, or other factors. In addition, the four-year graduation rate at four-year universities is only 43.6% [5]. The ability for students to determine, personalize, and revise the optimum graduation path efficiently based on individual capability, background, and future career interests could be a natural remedy to the aforementioned shortcomings. The original research contribution of this work is to address these issues by developing

an intelligent recommender system that pervasively personalizes and optimizes a student's path to graduation by accounting for the student's career interests and academic background, in addition to degree and institutional constraints. To this end, we plan to carry out the following tasks:

- Formulate the personalized recommender system that populates a student's graduation path. It is a multi-objective scheduling approach that both selects and schedules courses based upon several constraints. The objectives of this optimization problem are to minimize the number of semesters (time-to-degree), minimize the number of credit hours (tuition costs), and maximize the number of courses of interest (personalization). The outcome of this process is an ordered list of courses arranged by semester.

- Implement and validate the functionality of the proposed recommender system to ensure its effectiveness in developing a suitable trajectory path to graduation for an individual student. This will be done by testing a wide array of real student data with differing conditions, ranging from freshman to seniors, and running degree audits with the resulting output.

Uniqueness of our approach relative to previous efforts lies in the fact that previous course recommender approaches have solely focused on optimizing the graduation path of a student, ignoring the personalization aspects. The proposed approach is intelligent and dynamic, creates a more personalized experience allowing students to be more aware of their course load, potentially reduces stress, increases the university's retention rate, and potentially reduces the student's educational cost, while increasing success and graduation rates.

The remainder of this thesis presents the features of the intelligent recommender system, starting in Section 2 with a review of existing works in the field, including works in content-based personalization, path optimization, and elective personalization. Section 3 continues to review existing work in the field, specifically pertaining to the previous work done on the PERCEPOLIS project. With knowledge of existing work in mind, Section 4 details the foundation of the proposed intelligent recommender system and its various components. Section 5 presents the implementation and analysis of the results of the various test cases used to validate and test the system. Lastly, the thesis concludes with final thoughts and suggestions for future research in the area.

2. LITERATURE REVIEW

2.1. OVERVIEW

Personalized learning has been proposed as a means to improve a student's success, ultimately affecting both the retention and graduation rates of an institution. Personalized learning has been defined in many different ways [6], but Patrick *et al.* [7] puts it quite clearly: "Personalized learning is tailoring learning for each student's strengths, needs and interests –including enabling student voice and choice in what, how, when and where they learn –to provide flexibility and supports to ensure mastery of the highest standards possible". It has advanced in several directions: the recommendation algorithm approach [8], the path optimization approach [9], and the predictive approach [10]. The recommendation algorithm approach focuses on tailoring content and/or learning methods to the user. This approach is applicable on a broad scale, including grade schools, colleges, online course providers (like Khan Academy and Coursera), and workforce training systems. The approach is further divided into two areas: content-based recommendation and collaborative filtering recommendation. The path optimization approach focuses on optimizing a student's graduation path based upon prerequisites and a balanced course load. This area serves as the basis for this work and as such these works are closely related to this work. The predictive approach concentrates on the use of various predictive analysis tools for determining the likelihood of a student success, whether it be passing a course or finishing their degree. Figure 2.1 provides a visualization of these areas with the educational technology field, along with the papers that are associated with each area. This chapter is intended to discuss these approaches, enumerate and highlight their characteristics, and compare and contrast them against each other.

2.2. RECOMMENDATION ALGORITHMS

Recommendation algorithm approaches have become increasingly prevalent over the past few years due to the growing expanse of data that is available on the Internet. Recommendation algorithms can easily be found in streaming services, fast food drive-thrus, and even search engines. These approaches are designed to make meaningful recommendations to users based on their profile and the profiles of other similar users. The content-based approach is generally used for recommendation algorithms within streaming applications and other similar content-providing applications [11]. This method uses the user's content history to suggest additional similar content. The approach assumes that most user's have a specific type of content that interests them and relies on that assumption

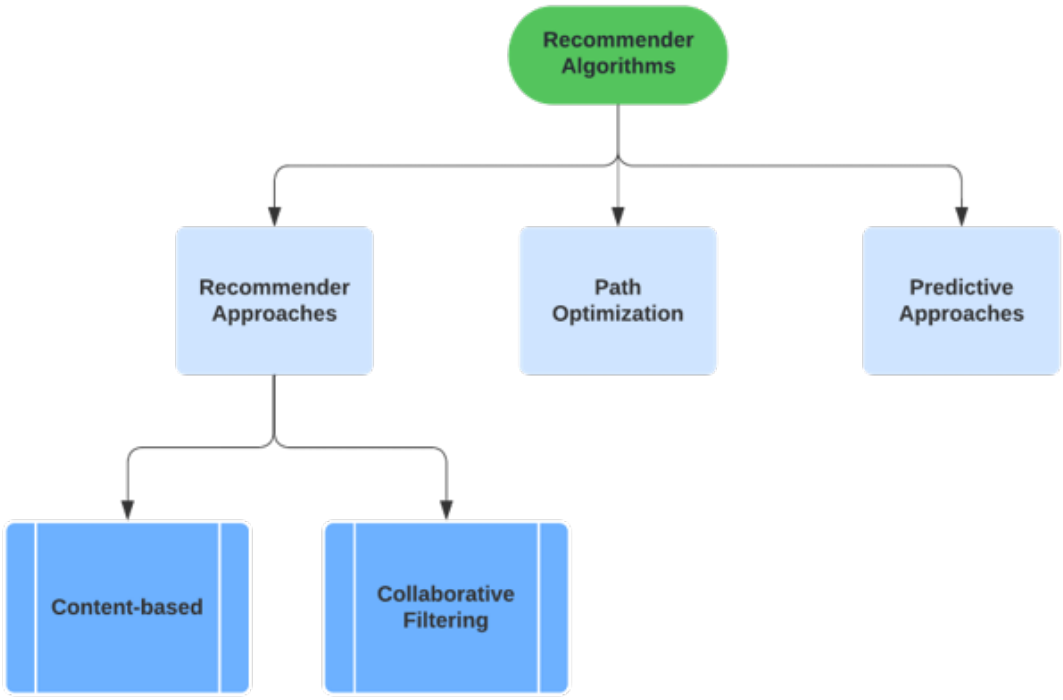


Figure 2.1. The classification of related works.

to provide accurate recommendations. Examples of this kind of recommendation algorithm can be found in the educational technology realm, typically within self-paced learning applications. On the other hand, the collaborative-filtering approach is commonly found in course recommendation algorithms. This approach aims to suggest content to users that other similar users have enjoyed in the past. The approach relies on having some sense of user's preferences or characteristics in order to succeed. A review of recommendation algorithms in e-learning, by Klašnja-Milićević *et al.* [12], provides a great definition of each of these approaches. Both of these approaches have been applied to course recommendation algorithms in different ways, some of which will be discussed in the coming sections. Due to the fact that content-based filtering relies upon historical user data and looks at the relationships between differing pieces of content, collaborative filtering has become the more popular method for content recommendation [13]. The next two subsections provide examples of both methods and explain how these examples differ from the approach proposed in this thesis.

2.2.1. Content-based Personalization. Content-based personalization is becoming an increasingly popular functionality across many different technical and learning applications. This form of personalization is currently more common in the self-paced learning sector rather than in academia. There are countless different applications available these days, but two of the dominating applications will be the main focus of the first part of this section: Khan Academy and Coursera. Both of these applications are web-based applications and are free to use in at least some form. Khan Academy is offered for students beginning at 1st grade and continuing up through general college-level topics [2]. Khan Academy is heavily focused on providing students with a free way to learn about different topics and offers a wide array of resources to help students in areas where they are struggling in. The application also offers resources for studying for standardized exams, including the Scholastic Aptitude Test (SAT), American College Test (ACT), and most Advanced Placement (AP) tests. As stated before, the application is completely free, and if the user creates a free account the application will be able to keep track of completed courses and progress in on-going courses. Additionally the free account provides suggestions on which courses should be taken next and even additional courses that the user may be interested in. Though the level of personalization is small and may seem trivial, but it does tailor the user's experience to what he or she is interested in and prevents the user from having to search for additional topics to learn about.

Coursera is also a web-based application [3]. It is tailored to the collegiate and online course industry, with college level students and adults in the workforce as the main audience. With Coursera, not all courses are free. The application offers the option of pursuing different types of certificates and degrees through various participating universities. There is a wide range of topics available for studying, although courses in technology and data analytics are the most popular ones. Coursera acts more as a professional development tool rather than the educational tool that the Khan Academy is. Coursera also offers a similar recommendation algorithm for account holders that suggests additional certificates and courses that the user may be interested in. Both of the aforementioned applications are similarly structured and use very similar recommendations; both tailor their user-interfaces to the user, displaying courses that the user may be interested in at the forefront. Examples of this kind of work are easy to find in the field, where this kind of personalization has been applied to various different applications, including workplace training systems and streaming apps like Netflix. An example can even be found in the form of a course recommendation algorithm, where the student inputs a course that they know about and in return receive a suggested set of similar courses [8]. This is just a simple example of the content-based approach being used in a course recommender, but

there have been various other algorithms that have implemented the same approach by taking user interests and/or learning ability into account. The literature review of recommendation algorithms in education, written by Urdaneta-Ponte *et al.* [14], refers to several different examples of both content-based and collaboration filtering algorithms, including the algorithm created by Borges and Stiubiener, which uses both user interests and learning ability as a means to suggest learning resources to the user [15]. These resources included audio, visual, and interactive resources. This algorithm was designed to help students take advantage of the learning methods that benefit them the most, with the goal of improving individual student success in a given class. The algorithm by Jetinai works very similarly, providing resource recommendations utilizing the user's learning style [16]. In contrast to the both of the previous examples, the work proposed in this thesis acts as a course recommendation algorithm that also optimizes the student's graduation path. While our work does employ the content-based recommendation approach through the use of student interests, unlike the previous algorithms, it moves to do more than just recommend elective courses to the student. It selects and places courses into the student's graduation path in a near-optimal ordering, helping the student identify their exact path to graduation. Khan Academy, Coursera and the other examples do not offer variation in how courses or certificates are completed because they are purely focused on content recommendation of courses that are self-paced and readily available all the time. In contrast, in an academic setting some courses are only offered certain semesters and others must be taken in a certain order, so the timing of course completion is the key to the student efficiently completing their degree.

2.2.2. Collaborative Filtering Personalization. As mentioned previously, collaborative filtering personalization has become quite popular in the educational technology realm. This is mostly due to its method of comparing a user to other previous users under the assumption that users with similar characteristics behave similarly. Provided that the classification of the user is correct, the resulting set of recommendations generally tends to be more personalized due to the deeper look at the users characteristics. Analysis of the algorithm proposed by Zhu *et al.* [17] gives a good idea of how this method can be used for course recommendation. Their Android-based recommendation algorithm makes suggestions to students about which courses they should use to fulfill their electives. The algorithm utilizes clustering and historical student data to place the user among a group of students with similar interests. From there, the algorithm recommends courses based upon which courses the students within the cluster took. The algorithm does well to incorporate personalization into the course selection process, but it does lack consideration of the student's career interests. This

could lead to incorrect clustering for a student who has an interest that is not currently reflected in their course history. This could lead to an improper recommendation and could lead to the student enrolling in courses that they are not interested in. Another education-based recommendation algorithm focuses on using user information and behavior to calculate similarity between users [18]. This algorithm then uses the user's learning ability to break any equivalences in similarity, allowing the algorithm to recommend e-learning services based upon the user's similarity to a specific set of previous users. This algorithm works very similarly to the one proposed by Zhu *et al.* [17], but works toward a different purpose. The algorithm proposed in [19] clusters students based upon their course history. This allows the algorithm to identify a user or set of users that is further along in their degree than the input user. The algorithm then makes suggestions to the user based upon the courses that the similar users took previously. Like the algorithm proposed in [19], the algorithm proposed in [20] creates clusters, but instead bases them upon basic student attributes, including GPA, age, ethnicity, and gender. Users are then placed into these clusters based upon these attributes and given course recommendations based upon the cluster that they were placed into. In conclusion, collaborative filtering heavily utilizes clustering methods by clustering users into classes of similarity based on different parameters. These algorithms all do well with providing recommendations, whether it be for learning resources or courses.

2.3. PATH OPTIMIZATION

Premalatha and Viswanathan's model focuses on optimizing graduation paths and providing a difficulty rating for each course in the path [9]. Their approach largely bases itself upon the original PERCEPOLIS implementation[21], as it utilizes prerequisites and a credit hour limit as constraints to the optimization problem. The approach also assigns difficulty ratings to each of the courses in the graduation path, which are then used to help balance the student's semester load. This approach is quite effective in its goal; however, it lacks personalization. In addition, difficulty is a subjective measurement: one student may consider a course extremely difficult while another may consider it to be easy. A similar system identifies itself as an elective recommendation algorithm, but aims to select sets of electives that will allow the student to perform well in their academic field [22]. While it is termed a recommendation algorithm, its goal of selecting a preferrable option may make it seem similar to an optimization problem. Despite the perceived similarity, this system does not have the components of an optimization problem and cannot claim to select the most optimal option. In contrast, our approach does optimize the graduation path and goes even further by populating and

personalizing the path. Selecting courses based upon relation to career interests and minimizing the addition of unneeded courses helps ensure the student is taking the courses that are the best for their graduation path and career. Another system, called Curricular Analytics, is an analytical system designed to identify bottlenecks and other potential areas of improvement within in a degree program [23]. The resulting analytics are designed to help academic departments redesign their degree programs in a way that promotes student success and retention. While this system aims to increase student success and retention just like our work, it approaches the problem in a different way. Our work focuses on creating an optimal graduation path based upon the degree requirements given, among other factors. The Curricular Analytics system focuses on restructuring the degree program itself in a way that makes it easier for students to navigate completing their degree.

The original implementation of the PERCEPOLIS project served as a basis for our work and dealt with path optimization as well [21]. This implementation took a two-step approach to populating graduation paths. The first step involved selecting courses for the student that allow them to meet their degree requirements. This step also attempted to fill the student's schedule with courses matching their interests. The second step then took the resulting set of courses and placed the courses into an optimized semester-by-semester ordering based upon the number of postrequisites that each course has. This method prioritizes putting critical path courses (courses that unlock numerous other courses) in order to shorten the student's time-to-degree as much as possible. The work in this paper combines the two steps from the original implementation into one, where courses are selected and placed into an optimal ordering at the same time. This allows for the best course, both schedule-wise and requirement-wise, to be recommended at any given point in the path.

2.4. PREDICTIVE APPROACHES

The educational technology field has also moved into predictive technologies over the past decade. These approaches are heavily focused on attempting to predict a student's probability of success. Different levels of success have been defined, including the probability of passing a given course and the probability of completing a degree. The system proposed in [10] used a regression process as the basis for prediction. It looks at courses individually and accounts for the difficulty of each course, the student's academic level, and the quality of the instructor. These various aspects allow the models to be more accurate due to the course and student-specific nature of the prediction. The predictive model proposed in [24] focuses on predicting whether a student will complete their degree, using machine learning to track and predict the student's performance within their degree

program. This approach takes the student's background and course selection into account, along with course-specific data such as the course's level of influence on the prediction. The system uses these parameters to predict the individual grades that the student will get in each course, which in turn allows it to predict the GPA that the student will have at the end of the term. There are several other works in this area of the educational technology field, and interested readers are referred to [25] for a comprehensive review of them. Finally the approach in [26] predicts a student's retention. This approach identifies several different measurements of engagement that can be retrieved from student portfolios to predict the student's retention.

2.5. SUMMARY

Overall, the works in the academic advising field are quite prevalent, but few have combined the course recommendation and scheduling process. According to the National Center for Education Statistics (NCES), only 43.8% of students at 4-year universities graduate within 100% of normal time (four years), while 62.4% graduate within 150% of normal time (six years) [5]. This means that about 19% of students take either five or six years to complete their degrees, which is likely due to taking unneeded classes, poor degree planning, or struggling in classes, among other reasons. Universities likely benefit off of this due to the increased revenue from additional tuition and possibly even additional room and board costs, so optimizing degree paths may not seem beneficial to a university. The truth is, improving the academic advising system does offer universities two important benefits: increased retention rates and higher 4-year graduation rates. These values are important to ranking companies, college students, and the reputations of universities.

3. PERCEPOLIS

The Pervasive Cyberinfrastructure for Personalized Learning and Instructional Support (PERCEPOLIS) program, of which this work is a part of, is a platform designed to facilitate personalized learning through an intelligent academic advising system. The foundation of PERCEPOLIS is to create an environment - through university systems - to allow students to create personalized course schedules and use the generated schedule as a base before discussing their next semester with an advisor. This chapter discusses the design of PERCEPOLIS and its expected effects on students when integrated into the university system.

3.1. DESIGN

In order for PERCEPOLIS to make informed decisions when making a student's personalized schedule, a variety of parameters must be received from multiple sources. The flow of data can be seen in the top-level diagram of Figure 3.1.

The environment's user interface obtains each student's preferences and ideal workload. These parameters, while defined by the student, may be overridden by an advisor or department head upon review of the student's schedule. Additionally, the environment will return warnings if certain inputs are found to produce unreasonable schedules. For example, if a student expected to graduate in only a few semesters but also set a low maximum credit hour count, PERCEPOLIS would indicate this discrepancy.

This information is then sent to the back-end containing algorithms for the course scheduler, graduation path planner, and course difficulty calculator. Our proposed approach fits into the PERCEPOLIS platform as the course scheduler. To generate the optimized semester schedule and graduation course path, the two linear programming algorithms also need data regarding the student's previous course history and the degree audit(s) from their given major(s). Given the optimized semester schedule, a clustering algorithm runs simultaneously given the student's past courses and grades to display a prediction of student success in the upcoming semester. When determining the functions of PERCEPOLIS, priority was given to the generation of personalized course paths. These semesterly schedules abstract away parameters like class meeting times, course availability, and instructor selection: elements that are subject to change and cannot be predicted more than one semester in advance.

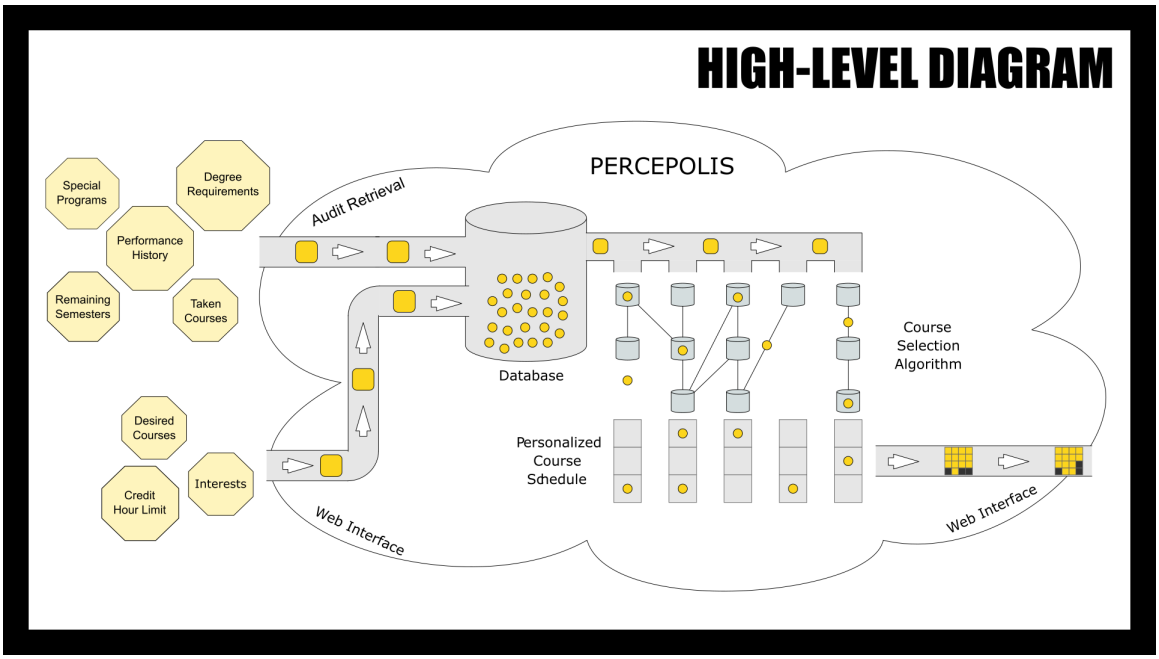


Figure 3.1. A high-level overview of PERCEPOLIS.

While the focus of PERCEPOLIS is primarily on the student experience, other users will also need to access the environment to perform other tasks related to student scheduling and advising. In order to accomplish this, different user types have been defined with different user intereface templates. These user types are Student, Advisor, Department Head, and Registrar. Along with inputting preferences and running the aforementioned algorithms, student users can view the course catalog and message their assigned advisor or department head. Advisor and department head users can, in turn, read and reply to messages sent from their respective students. This correspondence between faculty and students takes the form of requests and approvals/denials. The ability to add/delete and edit course parameters of the catalog used by the platform is commonly handled by the registrar's office. These users are given less interaction with the student experience and, depending on the existing structure of a university's pre-existing enrollment process, may find this feature useful to varying degrees. In the design phase of the user interface, these different user types had to be taken into consideration. Through the use of the Figma design software [27], the various interfaces for each user were laid out and routed to one another. The layout and routing is shown in Figure 3.2.

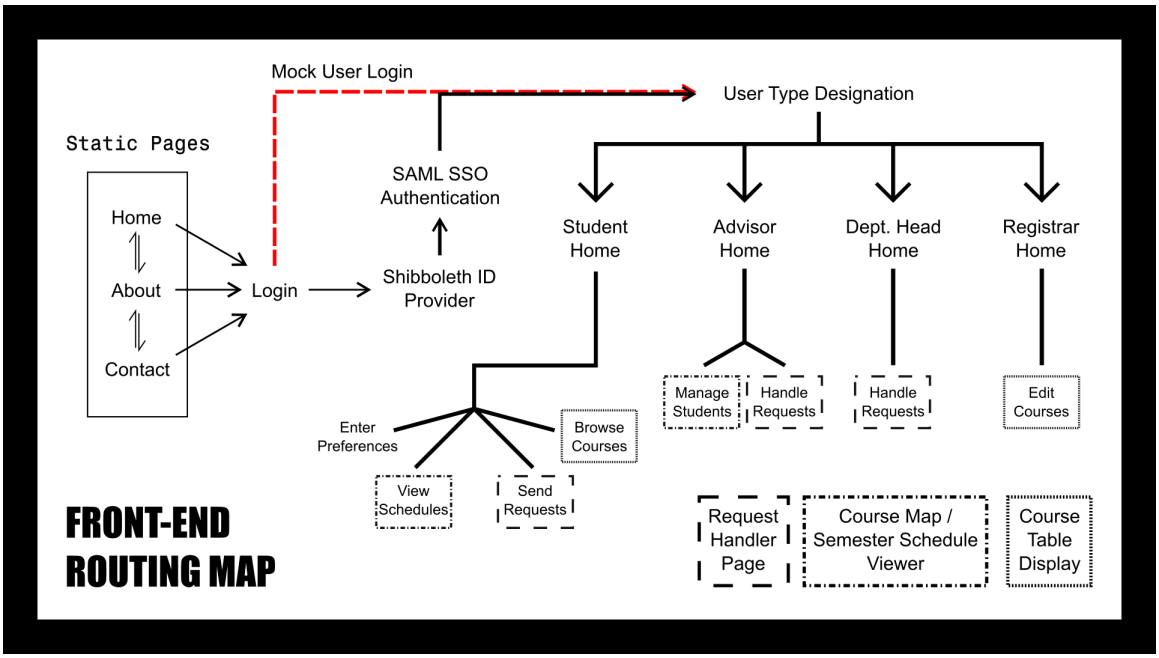


Figure 3.2. Front-end routing for each user type.

3.2. IMPLEMENTATION AND FLOW OF OPERATIONS

Displayed in Figure 3.3 is the existing environment for PERCEPOLIS. The front-end UI, which uses Angular 12 [28] and Typescript [29], sends the aforementioned student information to the back-end to be used as algorithmic input. The connection between front-end and back-end uses SpringBoot RESTful API [30]. SpringBoot was chosen because of its flexible libraries, fast performance, and helpful security features. For the back-end, all algorithms are written in Kotlin [31] and use PostgreSQL for the database [32]. This environment is not just for the purpose of running the two algorithms as described above. Instead, it's designed for perpetual algorithmic design and implementation for continued student success, satisfaction and retention. For the student, the PERCEPOLIS platform is, at its core, a course scheduling program created by three structures. 1) The client side; 2) The server side; and 3) the communication between them. A student will access the site and will be directed to the preferences page. From here, they will be asked a series of questions regarding their preferences for courses and pacing. This information will be sent to the Spring Boot back-end, with a RESTful API handling calls between the two. The back-end, given the data about the student, will then run a course scheduling algorithm and return the students recommended semester schedule and generated graduation path. Lastly, the existing environment

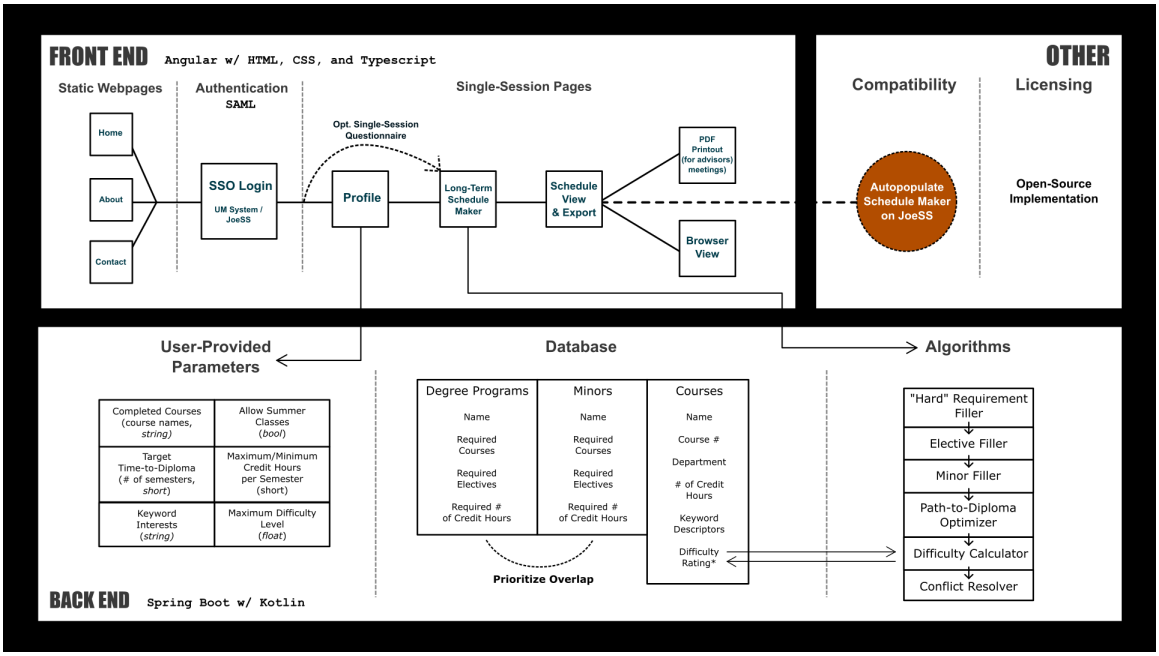


Figure 3.3. Functional diagram for the PERCEPOLIS implementation.

has been designed with compatibility and open-source licensing in mind. The application is designed to be able to easily retrieve information from other university systems, and is designed to be an open-source application for students to use for free.

3.2.1. Client Side. The client side was created using HyperText Markup Language (HTML), Cascading Style Sheets (CSS), Angular 12, and Typescript. HTML and CSS were used for their imperativeness in any web-based development. Angular 12 was used because of its component based architecture. Angular 12 has a wide selection of modules to choose from and allows the ability to quickly and reliably develop working pages seamlessly. In addition, the ability to organize page elements into components makes the implementation of widely-used features much easier. The component architecture allows the developers to save time in having to implement needed components for the web page by them already being implemented in the Angular structure. Furthermore, Angular 12 has the ability to be developed across multiple platforms including web, mobile web, native mobile and native desktop. It also manages http requests to the server. This allows the client to send a request and then consume the data given back by the back-end.

3.2.2. Communication Side. The communication side was created using SPRING Boot and a RESTful API. REST services have quickly become a de facto in building web services for its ease in building the services and consuming them. Furthermore, it offers many suitable actions including GET, POST, PUT, DELETE, caching and security (encryption and authentication). These features are built into the system and allow for ease of implementation.

RESTful API is also suitable with our back-end, which uses Kotlin as the language of choice. Additionally, its large-scale adoption means several online resources are available for understanding the framework and its implementation in specific languages. It also is fast and performance is a large priority for the developers of this framework.

3.2.3. Back-end Side. The back-end side was created using Kotlin. Kotlin is object-oriented and supports functional programming features. It is used for creating the database which will contain the data pieces needed for running the algorithms. The database system used is PostgreSQL. It is a free and open source software used for its proven architecture, reliability, data integration and robust feature set. It is highly extensible in its ability to define unique data types and build custom functions. These are significant as it allows the architecture to support custom data types needed for working with data such as courses, students and much more.

Kotlin is also unique in its ability to seamlessly create algorithms which are able to use the databases created within the system. Furthermore, these algorithms are able to be easily called with a service through the REST controller. This ease of use with the communication side made it ideal for its use in the PERCEPOLIS system.

3.3. ANTICIPATED OUTCOMES

The only current method for generating a course schedule is for students to manually create it using a degree audit. However, this approach is prone to fail in numerous ways, in no small part due to a lack of centralized information and the process itself being quite time consuming. Obtaining a personalized schedule that takes into consideration a student's interests, future goals, and desired course load requires the student to spend hours figuring out the description of each course. Even then, many students may miss a potential conflict or other available class that would suit them better. This ultimately leads to an unpersonalized and suboptimal schedule, increasing the chances that a student will find less success in their courses or be enrolled for a longer period of time than necessary. The PERCEPOLIS platform, which utilizes algorithms and pervasive computing techniques, will eliminate such uncertainties in the process.

At the university level, a personalized educational path could result in increased student satisfaction, an increased retentino rate, reduced time-to-degree, an improved graduation rate, and ultimately an increased student success rate. Other participants in the course selection process, such as advisors and registrar workers, will also benefit from the PERCEPOLIS platform through its streamlined course selection and registration, along with its improved student-faculty communication.

3.4. SUMMARY

This section was intended to give the big picture about the structure of the PERCEPOLIS platform and how this proposed course recommendation system fits into the PERCEPOLIS project. The platform is expected to evolve in the coming years, with additional functionality being added and the existing functions being improved. In all, the PERCEPOLIS project has given nine undergraduate scholars the opportunity to engage in research on a level that many undergraduates do not, and as a result has led to the formulation of various different avenues from which the project can continue to evolve and improve. The hope is that in the coming years these avenues will lead to better, more fruitful versions of the existing platform.

4. METHODOLOGY

4.1. OVERVIEW

As mentioned in Chapter 1, there were three main tasks that had to be completed in order to achieve accurate and useful results. These tasks included the formulation of the intelligent recommender system, the implementation and prototyping of the proposed system, and the testing and validation of the prosposed system. The proposed system is formulated and discussed in the following sections. The implementation of the system followed the same sections, with each section representing a small piece of the overall system. Lastly, the testing and validation process brought the pieces of the system together to ensure that every aspect of the recommender system worked properly. This work was carried out as a part of the PERCEPOLIS project within the Sendecomp research group, with four undergraduate students working on the project; collaboration with these students proved to be helpful when it came to design and implementation decisions.

4.2. SYSTEM DESIGN

The generation of the intelligent recommender system was an involved process due to the different aspects and constraints that need to come together to generate a valid output. Figure 4.1 shows the inputs and outputs of the system. Required inputs to the system include both the degree requirements, the student's transcript, course details, and course prerequisites and corequisites. The student also provides data such as his/her career interests and credit hour limit. The career interests are given in the form of keywords, assisting the system in selecting courses that the student is interested in. The interest-related courses are then used to fill as many degree requirements as possible, maximizing the student's college career experience by incorporating interesting and intriguing courses. A student can also enter their own semester credit hour limit in order to specify their pacing, or they can use the university's default limit. All of these inputs are combined to create a multi-objective optimization problem that output optimized graduation paths in the form of semester-by-semester schedules. The following subsections detail how optimization problem has been formulated.

The optimization component simultaneously selects and schedules courses into a near-optimal ordering so that the student can graduate as quickly as possible with their given pacing. The student is able to specify their own semester credit hour limit, allowing them to set a pace for themselves, or use the university's default limit. This module is formulated as a multi-objective optimization

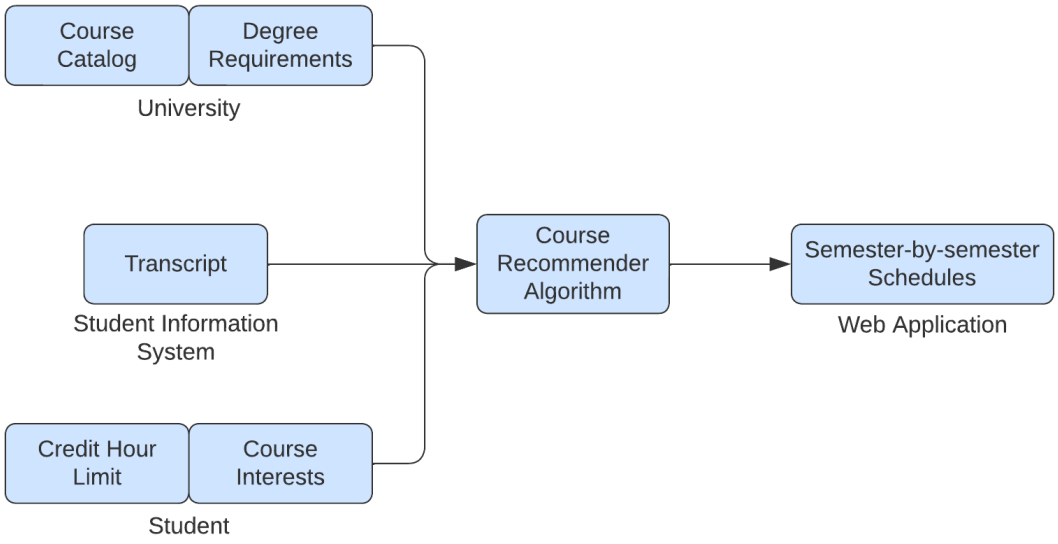


Figure 4.1. The inputs and outputs of the intelligent recommender system.

problem, with three objective functions, several different constraints, and a number of decision variables. The resulting output is a selection of courses organized into semesters, as shown in Figure 4.2. This figure offers a visualization of the optimization problem, including its constraints, decision variables, and objective functions, and shows how it is solved. The following sections detail the decision variables, constraints, and objective functions of the optimization problem. A list of symbols and notations used to formulate the proposed system is defined in Table 4.1.

4.2.1. Decision Variables. The decision variables are used to determine the set of recommended courses in a given semester. Course availability is determined based upon actual semester schedules (where available) and known availability patterns. The decision variables determine the recommendation of a given course in a given semester. Equation 4.1 formally defines the application of the decision variables. These sets of decision variables generate the result set for the optimization process, where each course in the set is recommended to be taken in the corresponding semester.

$$r_n = \{c : c \in S_n\} \, and \, r_n \subseteq S_n \tag{4.1}$$

Table 4.1. A list of symbols used in the optimization problem.

Symbol	Description
C	set of courses taken in previous semesters
CHL	student-set credit hour limit
CHR	total minimum credit hour requirement
H_p	number of credit hours taken in previous semesters
N	maximum number of semesters, acts as an arbitrary upperbound
P_n	set of prerequisites for courses in semester n
Q_n	set of corequisites for courses in semester n
R	set of all recommended courses
R_{int}	set of recommended courses that match student's interests
S_n	set of available courses for semester n
c	course
h_c	credit hour load for a course c
h_n	semester load in credit hours
n	semester
r_n	set of recommended courses for semester n

4.2.2. Reusability Constraint. Since a course is most likely available in multiple semesters, it is possible that a course could be repeatedly selected in subsequent semesters. While this is acceptable for repeatable courses, such as research courses, it is not valid for a large majority of courses. The constraint in Equation 4.2 ensures that unrepeatable courses are only taken once.

$$R = \bigcup_{n=1}^{N} r_n \; and \; \{c \in R\} = 1 \tag{4.2}$$

4.2.3. Credit Hour Constraints. The credit hour limit, as an input to the optimizer, sets a hard limit on the number of credit hours that can be placed into a given semester and hence restricts the student's load in a semester. The following constraint in Equation 4.3 ensures that the credit hour limit is never surpassed in a given semester.

$$h_n = (\sum h_c, \forall c \in r_n) \leq CHL \tag{4.3}$$

In addition to the semester credit hour constraint, many degrees require that a number of hours be completed for graduation. Equation 4.4 defines this parameter as a constraint for the optimization problem.

$$H_p + \sum_{n=1}^{N} h_n \geq CHR \tag{4.4}$$

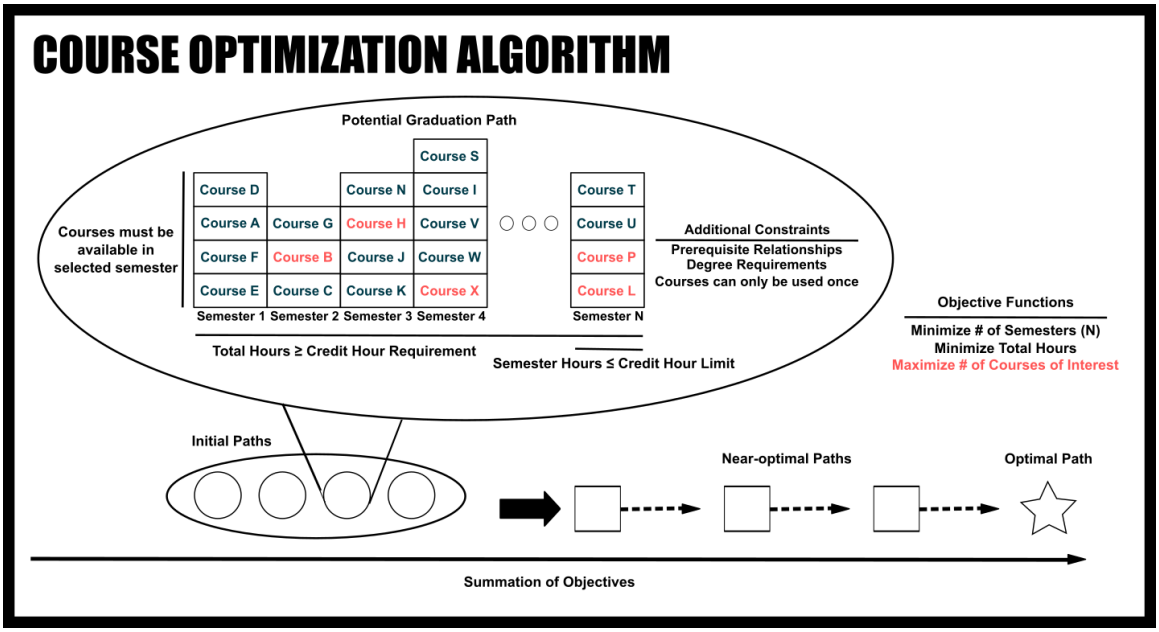


Figure 4.2. Architecture of the intelligent recommender system.

4.2.4. Degree Requirement Constraint. The degree requirements are the institutional constraints and need to be checked to ensure that the student is able to finish their degree. As formulated by Equation 4.5, as a part of the optimization process, they will be enforced by the optimizer.

$$degreeAudit(C \bigcup R) = True \tag{4.5}$$

The $degreeAudit()$ function represents the university's degree auditing system, which will return true for fulfillment of all requirements. It should be noted that miscellaneous requirements, such as GPA requirements, are assumed to be fulfilled by the student over the course of their college career.

4.2.5. Prerequisite Constraint. A large number of courses have prerequisites that must be completed prior to taking a course. Failure to take the prerequisites first results in the student being unable to take the course on time, resulting in a delay in graduation. Thus, it is very important that the optimization process takes this constraint into account for determining the sequence of course schedules (Equation 4.6).

$$P_n \subseteq \{\bigcup_{i=1}^{n-1} r_i\} \cup C \tag{4.6}$$

In short, Equation 4.6 ensures that all prerequisites for the courses in a semester have been taken prior to the given semester.

4.2.6. Corequisite Constraint. Some courses also have corequisites that must be taken either before or in the same semester. These courses are usually labs that are meant to be taken alongside a lecture-based course. This constraint is very similar to the prerequisite constraint, and is defined in Equation 4.7.

$$Q_n \subseteq \{\bigcup_{i=1}^{n} r_i\} \cup C \tag{4.7}$$

4.2.7. Optimization Criteria. There are three objective functions for the optimization process, each of which plays a role in generating an optimal solution. Two are identified as primary objectives, while the last one is a secondary objective. The primary objective functions aim to minimize time to degree in two ways: the first minimizes the number of semesters that the student must take, and the other minimizes the total number of hours that the student must take. Together, both of these ensure that the student does not take extra unneeded courses and the course are consolidated into as few semesters as possible. This minimizes time to degree and, depending upon the university's tuition structure, could minimize tuition costs as well. The primary objective functions are defined in Equations 4.8 and 4.9.

$$\text{Minimize } \{max(n) : r_n \neq \emptyset\} \tag{4.8}$$

$$\text{Minimize } \sum_{n=1}^{N} h_n \tag{4.9}$$

The secondary objective function is intended to enforce personalization. This function maximizes the number of courses of interest throughout the schedule so that the student maximizes the personal value of their degree by taking courses that progress them towards their desired career. Courses of interest are found by matching keywords defined by the student with keywords that have been associated with each course. This objective function is defined in Equation 4.10.

$$\text{Maximize } R_{int} : R_{int} \subseteq R \tag{4.10}$$

4.2.8. Solution. The decision variables, constraints, and objective functions described above form the multi-objective optimization problem. The problem is solved using the Google OR Tools CP-SAT solver, which was chosen for its ease of use, versatility, and capability to solve large-scale optimization problems [33]. The solver is discussed in detail in the next section. The three objectives are weighted and combined into one linear objective for the solver. The resulting output of the optimization problem is courses, placed into ordered semesters in a near-optimal ordering that demonstrates very close to the shortest path to graduation, as depicted in Figure 4.2.

4.3. IMPLEMENTATION

The implementation of the intelligent recommender system consisted of building a production-ready application that allows the system to output optimized graduation paths. The application is web-based and is extensively discussed in the PERCEPOLIS chapter.

The recommender system solves the multi-objective optimization problem using the Google OR Tools CP-SAT solver [33]. This solver was chosen because it is readily available from Google and is relatively easy to use for those new to optimization. Additionally, the CP-SAT solver is capable of handling a large number of constraints and decision variables, which makes it ideal for our application domain composed of a large number of decision variables and constraints. The Google OR Tools solver suite was also used in earlier works under the PERCEPOLIS project, most notably [21]. The solver is designed to be able to solve linear optimization problems through the use of the Large Neighborhood Search (LNS) method. The LNS method initially finds a valid solution and then iteratively selects a number of decision variables to adjust the solution in each round. The solver adjusts the selected decision variables, looking for the most optimal solution within the "neighborhood" that the solver has created. The neighborhood is defined by the set of decision variables the solver decides to adjust. A larger set of decision variables leads to a larger neighborhood. The solver utilizes large, changing sets of decision variables to ensure each neighborhood has enough differences from the last one, which allows the solver to avoid getting stuck in locally optimal solutions. Each round, the solver changes which set of decision variables it uses, changing the neighborhood in which it searches. This leads to either finding a more optimal solution or sticking with the previous solution each round. The solver stops searching for more optimal solutions once it reaches a timeout limit or if it begins keeping the same solution round after round. At this point, it has found a near-optimal solution and outputs the solution. The CP-SAT solver offers a wide range of options for defining constraints and allows constraints to be defined very clearly, so that it is easy for the user to ensure that they are

correctly defining their constraints. The solver offers a simple way of defining objective functions as well, with both minimize and maximize options. An important point to note is that the solver finds near-optimal solutions and does not always find the most optimal solution.

4.4. SUMMARY

Section 4 detailed the intelligent recommender system and its formulation as a multi-objective optimization problem. The section started with the formulation of the decision variables, constraints, and objective functions, which helped further define the optimization problem itself. From there, the discussion turned to the implementation of the recommender system, which included its integration into a web application that makes the system easy to use. The next section details the testing and validation plans, which includes test data and analysis of the generated output.

5. VALIDATION AND ANALYSIS

5.1. OVERVIEW

After the implementation of the proposed intelligent recommender system, the testing and validation process comes next to verify the validity of the approach. Several metrics were used to test validity, accuracy, and reliabilty, including the satisfaction of degree requirements and course relationships, the minimization of the semester and credit hour count, and comparison to the graduation path of a graduated student. The testing process included creating a database for three degree programs: computer engineering, computer science, and electrical engineering. The database contained all major courses, elective courses, relationships among courses, and degree requirements for each degree program. These degree requirements and relationships among courses were extracted from the Missouri S&T course catalog. A student database was also synthetically developed with varying degrees of diversity. The proposed recommender system was run for each case and the scheduling results were recorded and analyzed. These cases allowed for testing against both new students and in-progress students, as well as students with interests and without interests. These cases were used to be able to compare amongst each other and to the course catalog's example schedules that are listed for each degree. The results of each test case were validated by checking that the constraints were correctly enforced, especially degree requirement constraints and prerequisite constraints. The recommended schedules were then checked for inefficiencies (instances where making a change would shorten time to degree). The results of the testing process are discussed extensively in the coming sections.

5.2. TEST ENVIRONMENT

A test environment was built around the recommender system in order to facilitate the testing and validation. This environment consisted of building several data classes and then creating data objects with those classes to represent students, courses, semesters, degree requirements, and relationships among courses. This section addresses in detail the structure of each of the data classes, as well as the actual test data used.

5.2.1. Data Classes. Several data classes were constructed in order to facilitate the testing and operation of the recommender system. There are six classes in total: course, degree program, prerequisite, requirement, semester, and student. Each class has both an interface and an implementation. Both the interface and implementation parts of the class contain the same objects, so only the implementation structure, which is used by the recommender system, is shown.

5.2.1.1. Course. This data class houses all information related to a given course. Each course is represented several attributes, including the subject, course number, availability, prerequisites, and corequisites. The course's keyword is also included so that courses of interest can be found to enforce personalization.

- Course
  - $courseOfferingNumber$: unique identifier
  - $subject$: the course's subject
  - $courseNumber$: the course number
  - $keyword$: the course's keyword for interest matching
  - $prerequisites$: a list of prerequisites for the course, uses the Prerequisite class
  - $corequisites$: a list of prerequisites for the course, uses the Prerequisite class
  - $availableSemesters$: a list of semesters for which the course is offered, uses the Semester class

5.2.1.2. Degree program. This data class is designed to store all information for a given degree program. This includes the credit hour requirement, required courses, and elective requirements for the degree. Please note that this data class does not include any information related to GPA requirements or other non-course-related requirements.

- Degree Program
  - $programId$: unique identifier
  - $degreeName$: name of the degree, i.e. "Computer Engineering B.S."
  - $creditHourReq$: the overall credit hour requirement for the degree
  - $requiredCourses$: a list of courses that must be taken in order to complete the degree, uses Course class
  - $requirements$: a list of requirements that must be fulfilled for degree completion, uses Requirement class

5.2.1.3. Prerequisite. This data class stores prerequisite and corequisite information for each course. The same class is used for both prerequisites and corequisites because they both follow the same structure except prerequisites must be satisfied before a course is taken, while corequisites must be satisfied before or while a course is being taken.

- Prerequisite
  - $prereqId$: unique identifier
  - $associatedCourse$: the course that the prerequisite belongs to, uses Course class
  - $satisfactoryCourses$: a list of courses that are able to satisfy the prerequisite, uses Course class [1]

5.2.1.4. Requirement. This data class contains degree requirement information. Degree requirements are structured similarly to prerequisites; only one course in the list of satisfactory courses must be taken in order to satisfy the requirement.

- Requirement
  - $reqId$: unique identifier
  - $associatedDegree$: the degree that the requirement belongs to
  - $satisfactoryCourses$: a list of courses that are able to satisfy the requirement, uses Course class [2]

5.2.1.5. Semester. This data class stores semester information. It consists of the season and year of the semester, and is used to connect a season and year to the numerical semesters that the recommender system uses.

- Semester
  - $season$: semester season, either Fall or Spring
  - $year$: semester year

5.2.1.6. Student. The student data class stores student information that the recommender system receives as input. This includes the student's interest keywords, their desired credit hour limit, their degree program, and any courses that they've already taken.

[1] Only one course in the list must be taken to satisfy the prerequisite.
[2] Only one course in the list must be taken to satisfy the requirement.

- Student
  - $studentId$: unique identifier
  - $degreePrograms$: a list of all degree programs that the student is currently working on, uses Degree Program class
  - $creditHourLimit$: the student's desired credit hour limit
  - $interestKeywords$: a list of interest keywords that have been input by the student
  - $takenCourses$: a list of all courses that the student has already taken, uses Course class
  - $completedHours$: the number of credit hours that the student has completed

5.2.1.7. Test data. The test data that was used for testing and validating the recommender system was quite extensive. The data included three degree programs (Computer Engineering, Computer Science, and Electrical Engineering) and all associated requirements and required courses, a number of semesters from 2022 up to 2030, and a wide range of courses. The courses included in the test data consist of all courses from the Computer Engineering, Computer Science, Electrical Engineering, and Math departments, and any other courses necessary to provide sufficient room for variation in schedules. In total, over two hundred courses were made available to the recommender system, which proved to be more than enough to see variation in the recommended elective courses. Two interest keywords were implemented for testing: a gaming interest, which includes courses in animation and video game technology, and a computer architecture interest, which includes courses related to computer design and architecture. Table 5.1 shows the courses that fall under each keyword. The full course list, along with the degree requirements for each degree, can be found in the Missouri S&T course catalog [34]. Note that not all courses listed in the catalog are a part of the test data.

5.2.2. Test Cases. A number of test cases were applied to the recommender system, with thirteen of them being presented in this thesis. Three of them act as base cases, which the remaining ten are variations of. Each base case belongs to one of the selected degree programs (Computer Engineering, Computer Science, and Electrical Engineering). These base cases allow for validation of degree requirements, prerequisites, corequisites, and credit hour requirements. As such, these cases use a freshman student that has received entry credit for the basic math courses (Math 1140 and Math 1160), a credit hour limit of 18 hours, and no defined interest. This credit hour limit of 18 was chosen because Missouri S&T requires approval to take any more than 18 credit hours in one semester. The base cases will also serve as control cases for the remaining cases. It should be noted that all remaining cases assume that the student has received entry exam credit for the basic math

Table 5.1. A list of keywords and their related courses.

Interest	Related Courses
Gaming	CS 5404
	CS 5405
	CS 5406
	CS 5407
	CS 5408
Computer Architecture	CpE 5110
	CpE 5120
	CpE 5130
	CpE 5151
	CpE 5160
	CS 3100
	EE 3100

Table 5.2. The first nine test cases.

Test Case	Degree Program	CHL	Interests	Course History
1	Computer Science	18	-	-
2	Computer Science	13	Gaming	-
3	Computer Science	19	-	-
4	Computer Science	16	-	-
5	Computer Engineering	18	-	-
6	Computer Engineering	15	Computer Architecture	-
7	Computer Engineering	19	Computer Architecture	-
8	Electrical Engineering	18	-	-
9	Electrical Engineering	19	-	-

courses unless specified otherwise. The remaining test cases were designed to demonstrate that the system optimizes the student's graduation path according to the three objective functions. Out of these ten cases, there are two interest-based cases, three cases that represent students at different levels of completion, three cases with varying credit hour limits, and two cases that are combinations of the previous three case types. The interest-based cases are designed to show that the system maximizes the number of courses of interest when the student provides at least one interest keyword, while the cases with varying credit hour limits are designed to prove that the system minimizes time-to-degree based upon that limit. The cases with students at different levels of completion are used to show that the system is able to compute graduation paths with differing levels of complexity. Lastly, the two combination cases prove that each of these individual cases can be combined and

Table 5.3. The remaining four test cases.

Test Case	Degree Program	CHL	Interests	Course History
10	Electrical Engineering	18	-	English 1211 English 1212 Math 1214 Math 1215
11	Electrical Engineering	18	-	English 1120 English 1160 Math 1214 Math 1215 Fr Eng 1100 Econ 1100 History 1200 Chem 1310 Physics 1135 Mech Eng 1720
12	Electrical Engineering	16	-	English 1120 English 1160 Math 1214 Math 1215 Fr Eng 1100 Econ 1100 History 1200 Chem 1310 Physics 1135 Mech Eng 1720
13	Electrical Engineering	18	-	English 1120 English 1160 Math 1214 Math 1215 Math 2222 Math 3304 Fr Eng 1100 Econ 1100 History 1200 Chem 1310 Physics 1135 Physics 2135 Physics 2305 Mech Eng 1720 CS 1500 CpE 2210 EE 2100 EE 2120 EE 2200

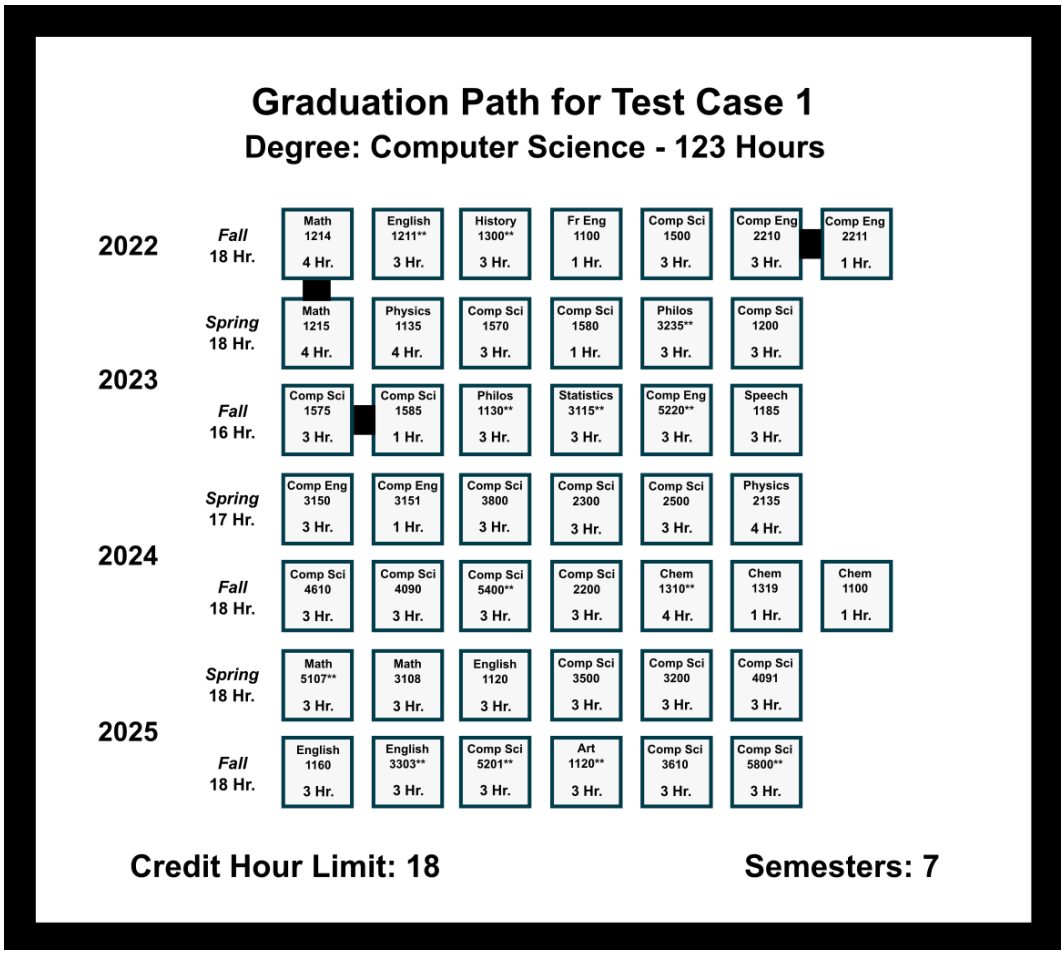
optimal results will be achieved. Tables 5.2 and 5.3 enumerate these cases and their respective parameters. Overall, these test cases are collectively designed to show the capabilities and flexibility of the proposed recommender system.

5.3. TEST RESULTS

The analysis of the generated graduation path for each case proved the validity of the proposed system. The recommender system found an optimal solution for each case and successfully generated graduation paths that fulfill all prerequisites and degree requirements. Additionally, the system runs quite quickly, taking under 30 seconds to compute a result on a standard laptop in most cases. Collectively, the test cases were able to prove that the recommender system fulfills all three objective functions: minimizing time-to-degree, minimizing credit hours, and maximizing courses of interest. Further explanation of the results of each test case is below.

5.3.1. Test Case 1. The first test case, shown in Figure 5.1, demonstrates the system's ability to output a valid graduation path for a new student in the Computer Science program. This is a base case for computer science program with an 18 credit hour limit. The template graduation path from Missouri S&T can be found in Figure A.1. In comparison, it is clear that the recommender system was able to minimize time-to-degree by maximizing the number of hours in each semester, which results in only 7 semesters being needed for graduation. The template graduation path does not maximize the number of hours in each semester, which results in the student taking a full 8 semesters to graduate. A simple way to check for the minimum number of semesters required to graduate includes calculating Equation 5.1. It is important to note that the student has received 5 hours of credit already due to the basic math courses, so only 123 hours are required to graduate. In this case the result is 6.83, meaning that at least 7 semesters must be taken before graduation is possible. It should be noted that this simplified equation does not consider fulfillment of all requirements, nor the length of the critical prerequisite path, which could require that additional semesters be taken. That being said, this equation does suffice for checking whether the recommender system has minimized time-to-degree. In this case, the path contains 7 semesters and 123 hours of credit, which matches the minimum number of semesters and hours required for graduation. It should be noted that the student did not express any interests, so there was no opportunity to maximize the number of courses of interest in the schedule.

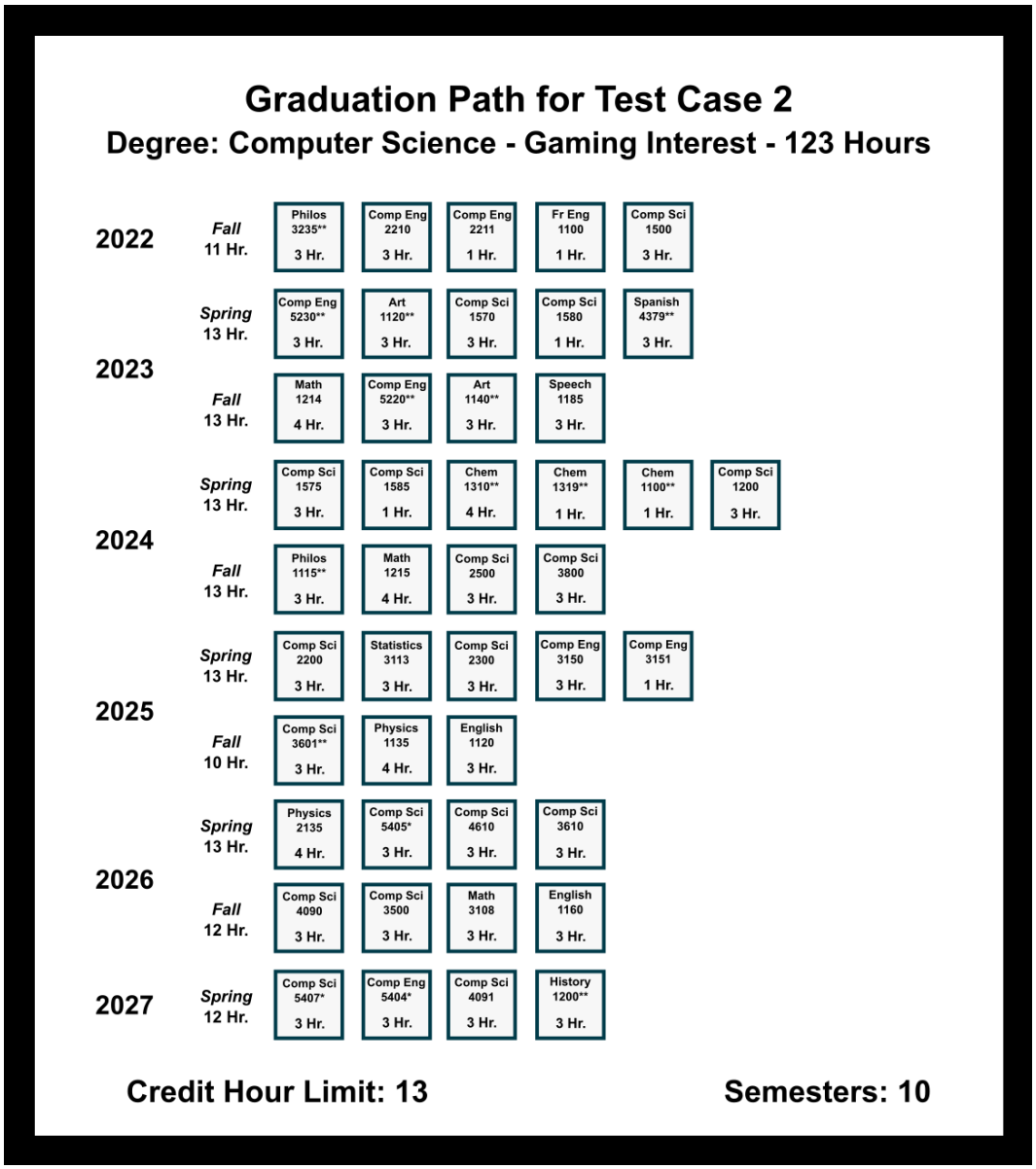
$$CHR - Completed\,Hours/CHL = Minimum\,\#\,of\,semesters\,possible \tag{5.1}$$



**denotes elective courses. All other courses are explicitly required.
Black boxes denote prerequisite (vertical) or corequisite relationships (horizontal). Only some relationships are shown, many more exist.

Figure 5.1. Recommended graduation path for the test case 1.

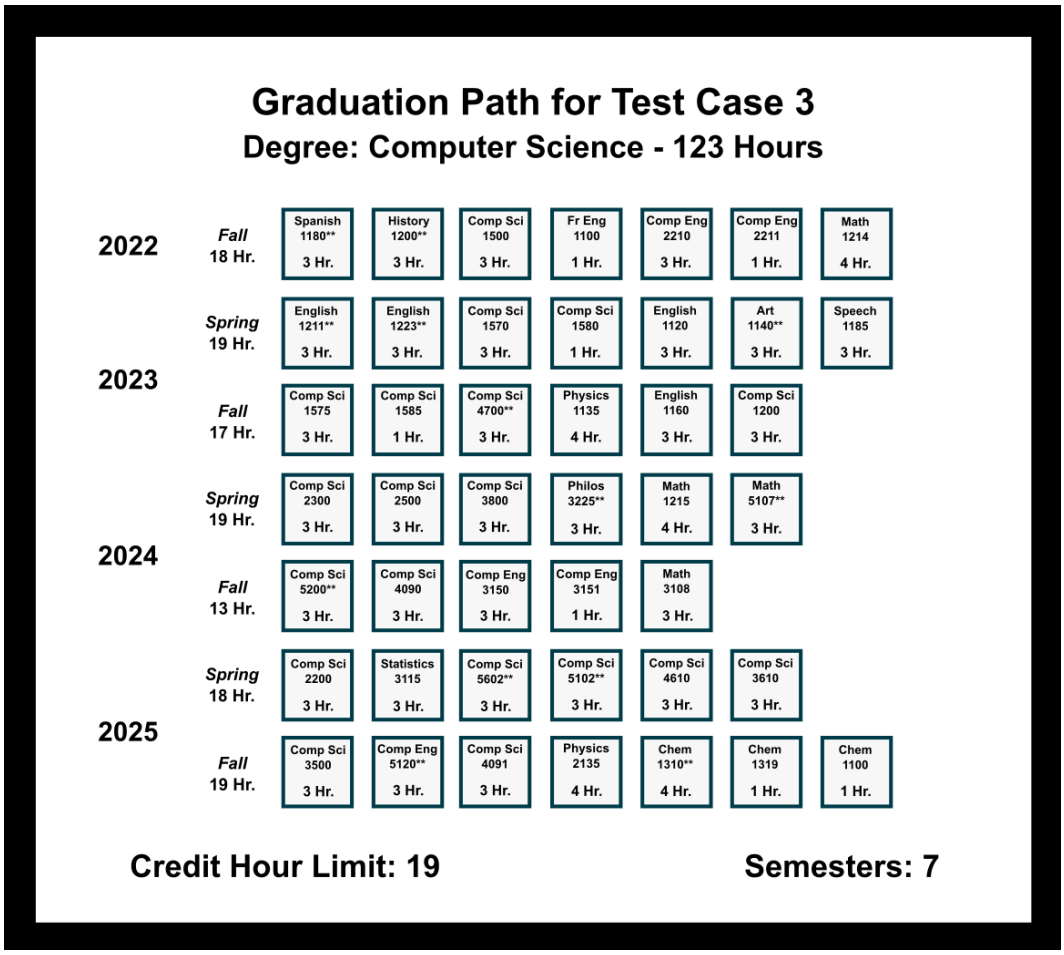
5.3.2. Test Case 2. The second test case (Figure 5.2) is similar to the first case, with the difference of expressing interest in "gaming" courses and using a 13 hour limit. As can be seen, the resulting graduation path includes 10 semesters and the minimum of 123 credit hours. Since the credit hour limit changed, the result from Equation 5.1 shows 9.46, meaning that the system has minimized time-to-degree. In this path, the system incorporates three courses of interest into the student's path: CS 5404, 5405, and 5407. Since all of the courses of interest for this particular interest are in computer science, this is the maximum number of courses that the system can use due



*denotes elective courses that match the student's interests.
**denotes elective courses. All other courses are explicitly required.

Figure 5.2. Recommended graduation path for the test case 2.

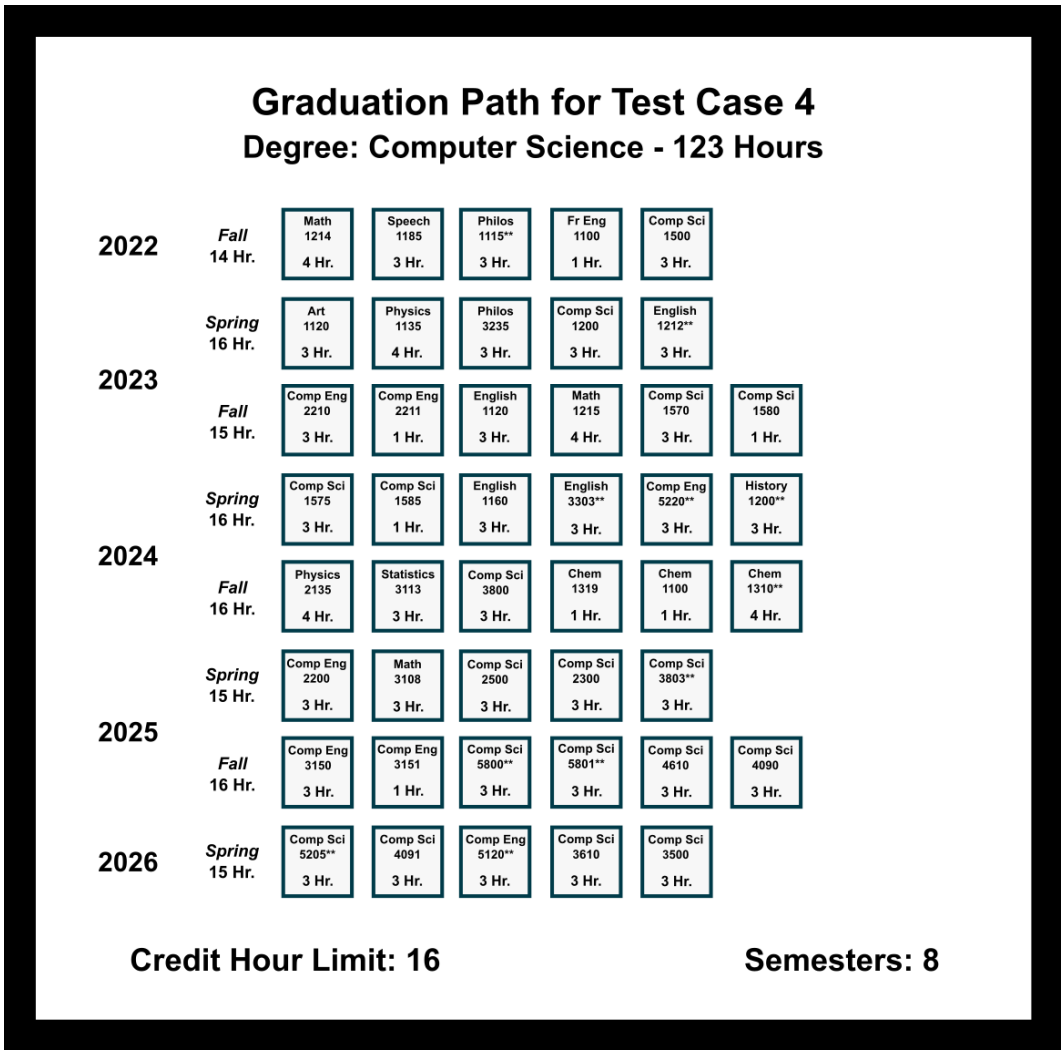
to the degree requirements. The computer science degree only allows for three 5000 level elective courses from the computer science department. Even with this constraint, the system still maximized the number of courses of interest and provided an optimized graduation path.



**denotes elective courses. All other courses are explicitly required.

Figure 5.3. Recommended graduation path for the test case 3.

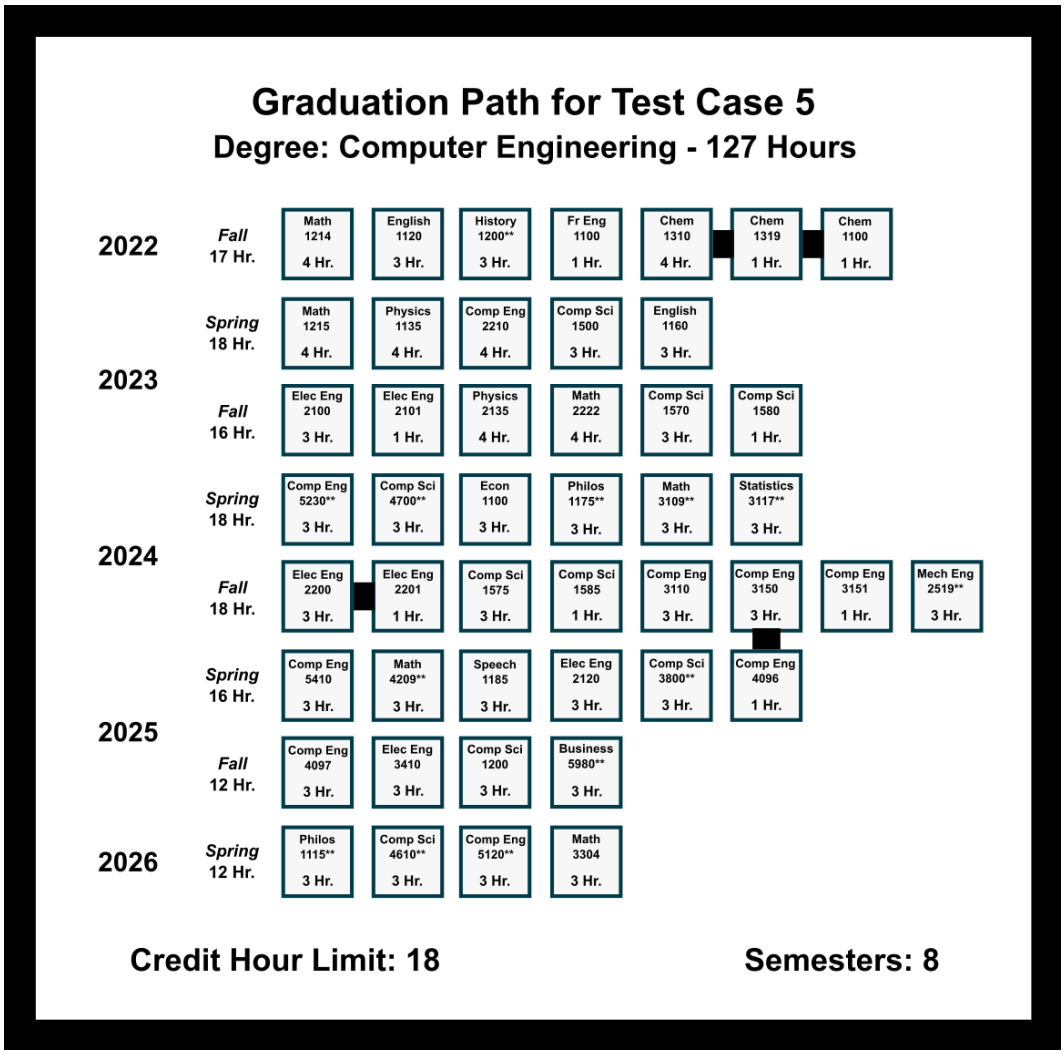
5.3.3. Test Case 3. The third test case (Figure 5.3) is also similar to the first test case, except the credit hour limit has been increased to 19 hours. Due to this change, the result to Equation 5.1 is now 6.47, meaning that despite increasing the limit the minimum time-to-degree is still 7 semesters. In comparison, the recommended path contains 7 semesters and 123 hours, both of which match their respective minimums. The biggest difference between the first case and this case is the variation in the number of hours in each semester (i.e. 13 and 19 per semester). This shows that when increasing the credit hour limit does not reduce the time-to-degree it could result in a wider range of hours in each semester. This is not inherently negative, as some students may prefer to have lighter semesters later in their path in order to allow for career preparations and planning.



**denotes elective courses. All other courses are explicitly required.

Figure 5.4. Recommended graduation path for the test case 4.

5.3.4. Test Case 4. The fourth test case (Figure 5.4) does the opposite of the previous test case and decreases the credit hour limit to 16 hours. This means that the result to Equation 5.1 becomes 7.68, and hence this student will graduate in a minimum of 8 semesters. As is shown, the recommender system outputs a path that contains 8 semesters and the minimum 123 hours. This further shows that whether the credit hour limit is increased or decreased, the recommender system will find a path that optimally minimizes time-to-degree while still conforming to the university requirements (degree requirements and prerequisites) and the student's parameters.

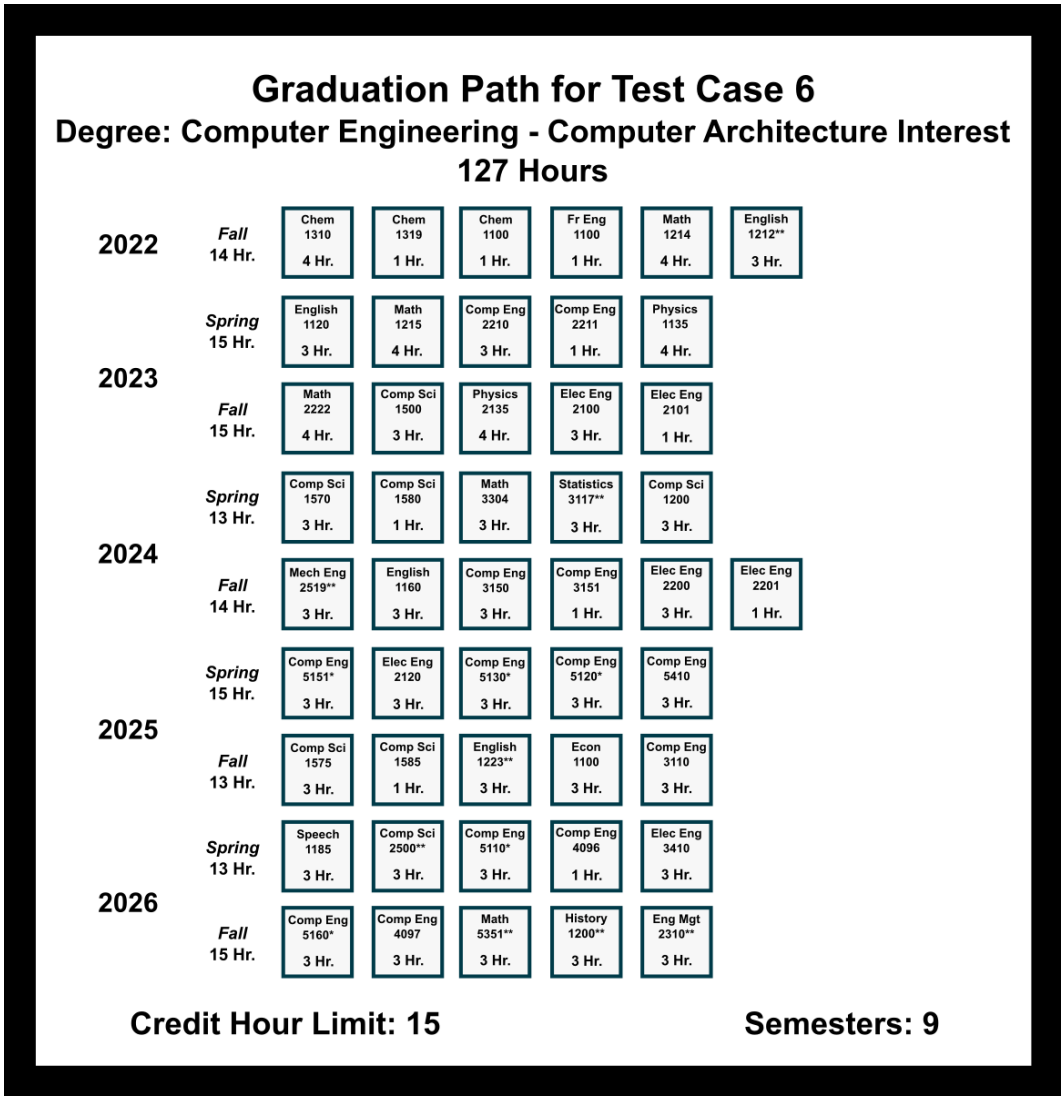


**denotes elective courses. All other courses are explicitly required.
Black boxes denote prerequisite (vertical) or corequisite relationships (horizontal). Only some relationships are shown, many more exist.

Figure 5.5. Recommended graduation path for the test case 5.

5.3.5. Test Case 5. The fifth test case, Figure 5.5, serves as the second base case, this time for the Computer Engineering program. As with the first base case, the credit hour limit is 18 hours and the student has not specified any interests. The minimum credit hour requirement in this case is 127. The result from evaluating Equation 5.1 on this case is 7.05, meaning that a minimum of 8 semesters must be used to complete the degree. The result of the recommender system is a graduation path that matches both of these minimums: 8 semesters and 127 hours. As a result, this path is also optimal. Like the first base case, this case was designed to be compared to the example graduation path provided in the Missouri S&T course catalog. The template graduation path can be

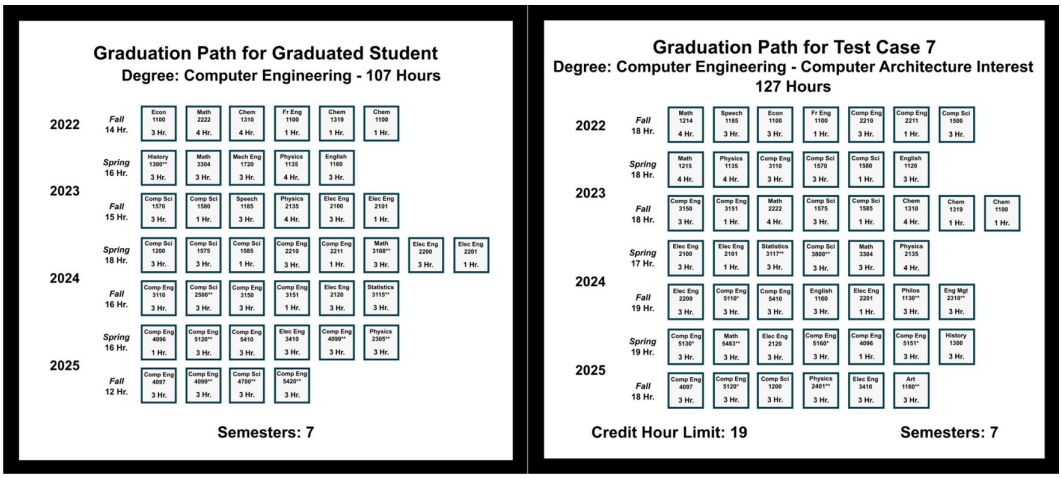
found in Figure A.2. These two paths are similar in that they both suggest completion in 8 semesters, but the recommender system only suggests 127 hours for completion while the example path suggests 128 hours for completion. This difference is due to the example path failing to consider that students get credit for the two basic math courses and as a result those hours contribute toward the 128 hour requirement. As a result, the recommender system is able to reduce the path by one hour and save the student on unnecessary tuition costs.



*denotes elective courses that match the student's interests.
**denotes elective courses. All other courses are explicitly required.

Figure 5.6. Recommended graduation path for the test case 6.

5.3.6. Test Case 6. The sixth test case (Figure 5.6) is the same as the previous one except the student expresses an interest in computer architecture courses and sets the credit hour limit to 15 credit hours. Due to the change in the credit hour limit, the minimum number of semesters is now 9, but the minimum number of hours needed is still 127. Just like the previous case, the recommender system outputs a path that matches these minimums. In contrast to the previous case, the recommender system selects five computer architecture courses as electives. The courses of interest that were selected are: CpE 5110, 5120, 5130, 5151, and 5160. These five courses are the maximum that can be put into the path because the other courses that match the interest do not meet the 5000 level requirement for the free elective. Since the recommender system prioritizes time-to-degree before courses of interest, it will not put unneeded courses into the path at the expense of adding additional semesters or hours.



*denotes elective courses that match the student's interests.
**denotes elective courses. All other courses are explicitly required.

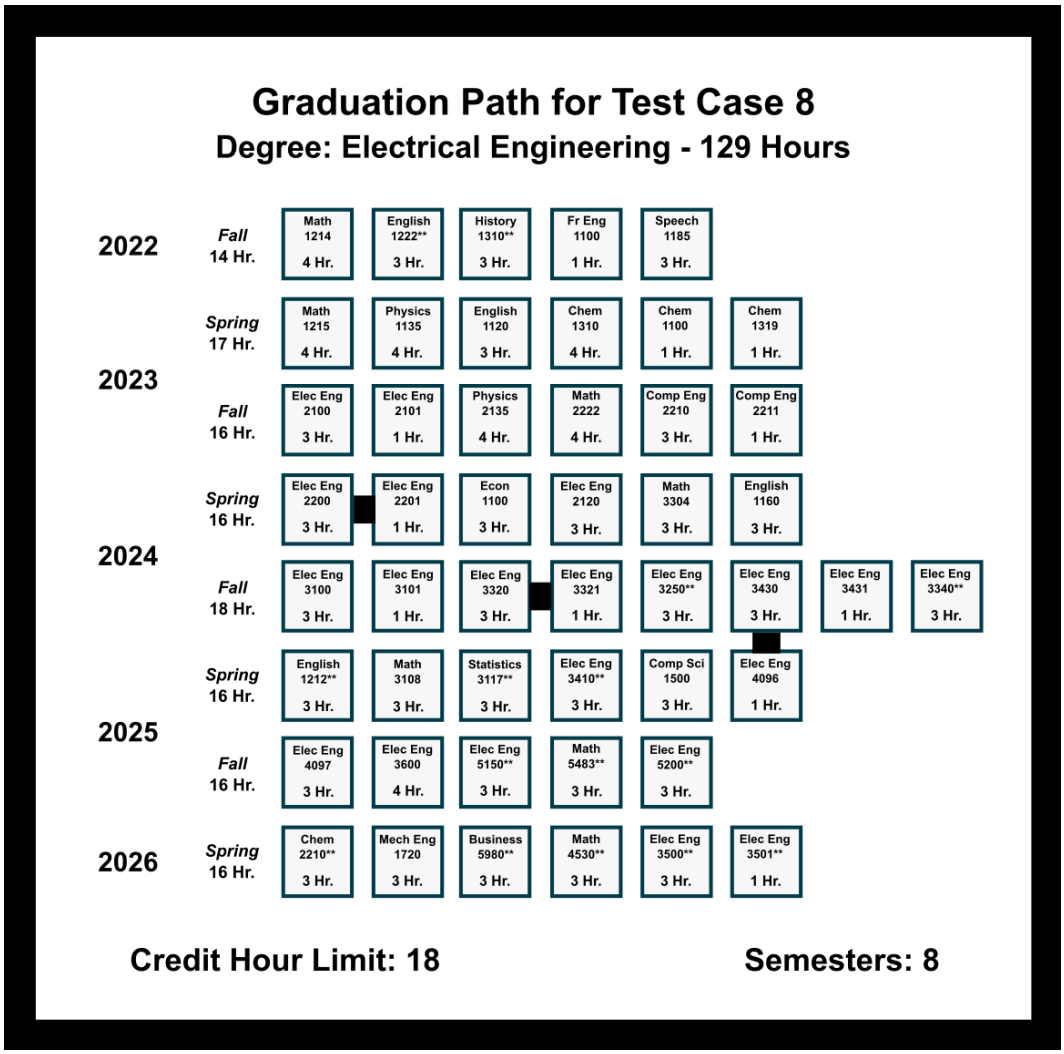
Figure 5.7. (a) Graduation path for a graduated student (left) and (b) the recommended graduation path for the test case 7 (right).

5.3.7. Test Case 7. The seventh test case (Figure 5.7b) is similar to the previous one, but instead increases the credit hour limit to 19 hours. This means that the result for Equation 5.1 is 6.68 and a minimum of 7 semesters is needed to complete the degree. The recommender system's output graduation path includes 7 semesters and 127 hours. The path also includes the same five courses of interest as in the previous case. In comparing to the minimums, this path is optimal and it includes as many courses of interest as possible. In comparison to the previous case, this shows that when the student increases the credit hour limit, the system will provide them with a path that

shortens the time-to-degree as much as possible. The graduation path found in Figure 5.7a is the actual graduation path for a student that has completed their degree. This student graduated in 7 semesters and took 129 credit hours in total, with 22 credit hours coming from AP credit taken in high school. Looking at the student's computer engineering elective courses, they took six hours of research, CpE 5120 (computer architecture), CpE 5420 (networks), and CS 4700 (intellectual property). These courses are quite diverse, each one comes from a different area of the computer engineering field. Comparing to the test case, the selection of courses is arguably random. If the student used the proposed recommender system and expressed an interest in networks or computer architecture, they would have received recommendation for taking courses in those areas rather than a random set of courses.

5.3.8. Test Case 8. The eighth test case, Figure 5.8, serves as the third base case, this time for the Electrical Engineering program. As with the first base case, the credit hour limit is 18 hours and the student has not specified any interests. The minimum credit hour requirement in this case is also 123, but degree requirements result in a minimum of 129 hours being needed. The result from evaluating Equation 5.1 on this case is 7.17, meaning that a minimum of 8 semesters must be used to complete the degree. The result of the recommender system is a graduation path that matches both of these minimums: 8 semesters and 129 hours. Like the other two base cases, this case was designed to be compared to the example graduation path provided in the Missouri S&T course catalog. The template graduation path from Missouri S&T is shown in Figure A.3. In this case, these two paths are similar in that they both suggest completion in 8 semesters, but the recommender system suggests 129 hours for completion while the example path suggests 128 hours for completion. This difference is due to the example path failing to consider that students must take the chemistry lab safety course, which counts for one hour. Unfortunately, this makes the resulting path appear to fail at minimizing the total number of credit hours, but in reality the example path fails to recognize a crucial prerequisite.

5.3.9. Test Case 9. The ninth test case (Figure 5.9) is similar to the previous one, but instead increases the credit hour limit to 19 hours. This changes the result from Equation 5.1 to 6.79, which means that at least 7 semesters are needed in order to fulfill the credit hour requirement. As in the previous example, the student needs 129 hours in order to fulfill all of the degree requirements. The recommender system produces a path with 7 semesters and 129 hours, matching the minimums

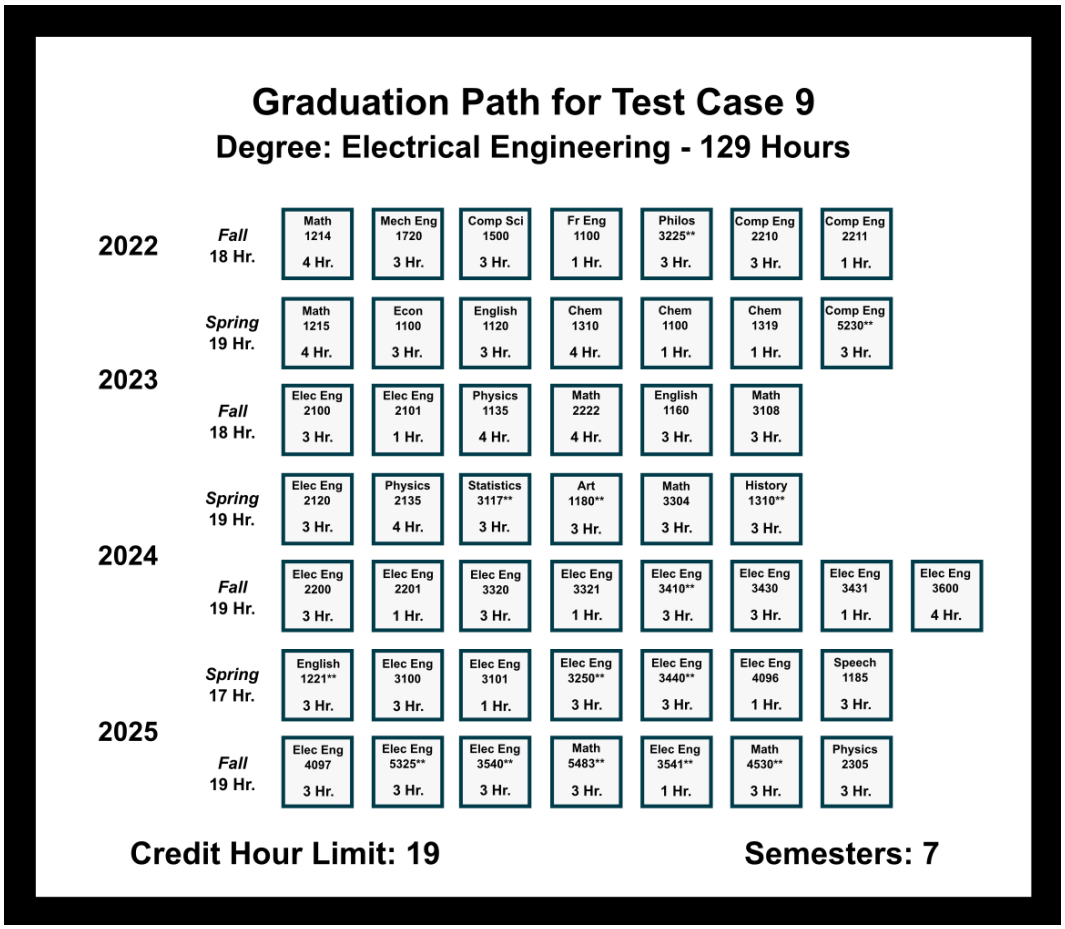


**denotes elective courses. All other courses are explicitly required.
Black boxes denote prerequisite (vertical) or corequisite relationships (horizontal). Only some relationships are shown, many more exist.

Figure 5.8. Recommended graduation path for the test case 8.

that were previously defined. Like case 7, this case proves that when the student increases their credit hour limit the recommender system responds accordingly and looks for the most optimal graduation path.

5.3.10. Test Case 10. The tenth case, Figure 5.10, uses a new student that has received AP credit for four courses from their high school. This reduces the number of hours that they need in order to complete their degree to 112 hours. As a result, with the 18 credit hour limit in mind, the result of Equation 5.1 is 6.22, meaning there is a 7 semester minimum. The recommender system's output graduation path is 7 semesters long and has a total of 112 hours, so the resulting output

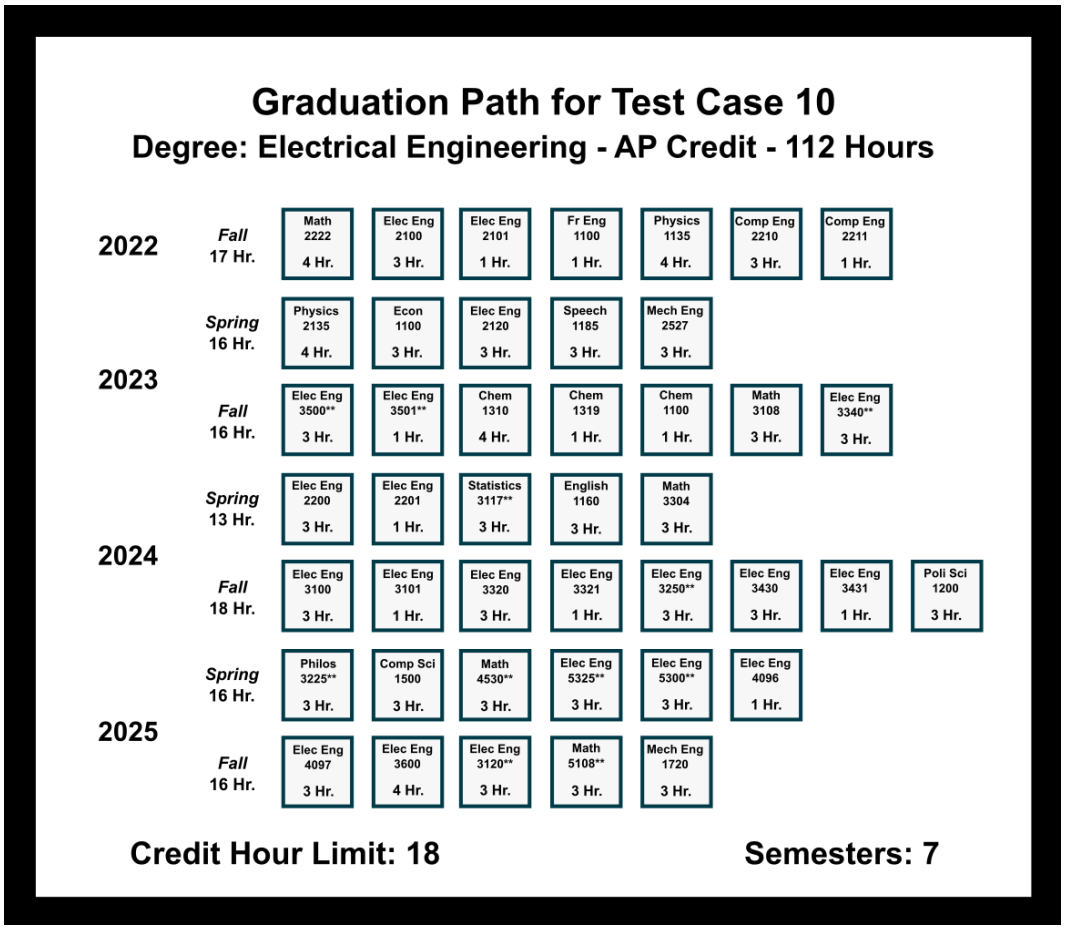


**denotes elective courses. All other courses are explicitly required.

Figure 5.9. Recommended graduation path for the test case 9.

is optimal. In comparing this case to its respective base case, it is clear that even a little bit of AP credit from high school can reduce time-to-degree by a whole semester. More importantly, this case takes a first step in showing that the recommender system can take students from any level of completion and will still give them an optimal graduation path for the remainder of their degree.

5.3.11. Test Case 11. This test case (Figure 5.11) continues the demonstration that the recommender system can take any level of student as input. This case uses an incoming sophomore who has already completed a number of courses during their freshman year. The student's course history can be found in Table 5.2. It should be noted that these courses were chosen based upon the suggested freshman schedule in the example path for the Electrical Engineering program. Since the student has already completed one year of their path, the recommender system only needs to compute the remainder of their path. The student only needs 95 more hours in order to complete their degree.

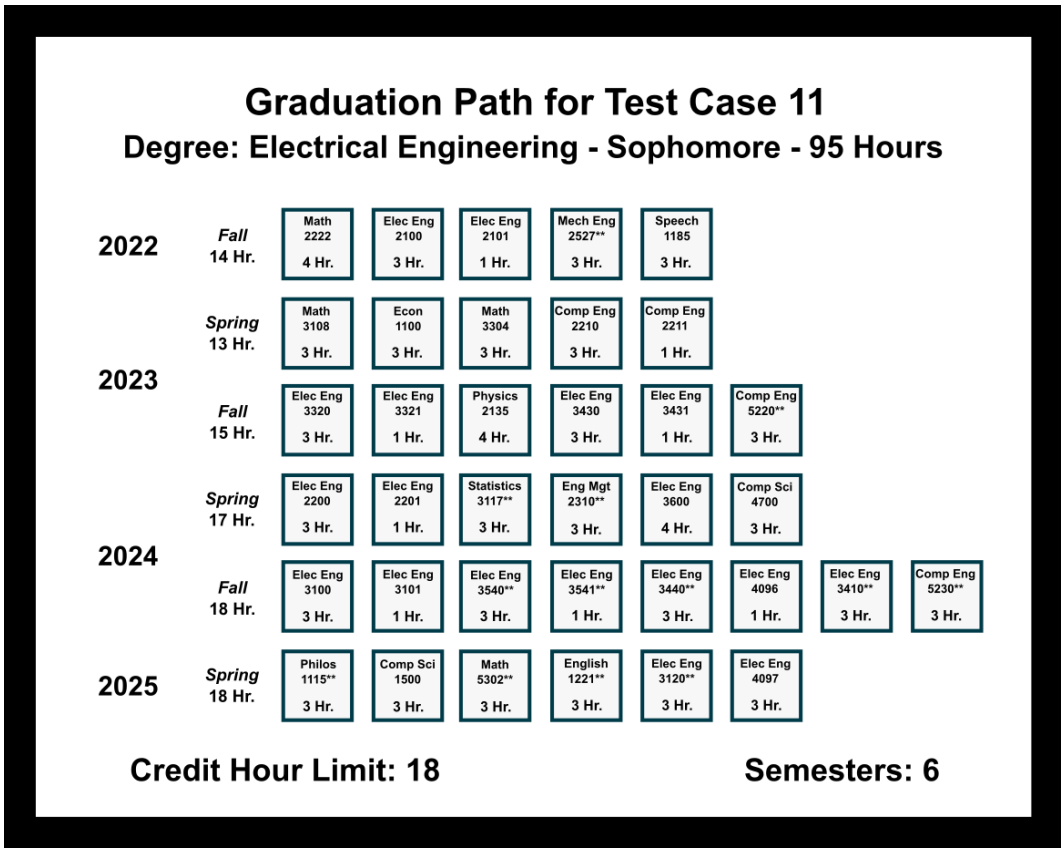


**denotes elective courses. All other courses are explicitly required.

Figure 5.10. Recommended graduation path for the test case 10.

This case uses an 18 credit hour limit, which means the result of Equation 5.1 is 5.28. Therefore at least 6 semesters are required to complete the degree. The path generated by the recommender system has 6 semesters and 95 hours, which matches the computed minimums, meaning that the result is optimal. This case further proves that the recommender system is able to generate optimal graduation paths for any type of student, regardless of which courses they have previously completed.

5.3.12. Test Case 12. This test case (Figure 5.12) is designed for comparison to the previous one. The same student is used, except the student specifies a 16 credit hour limit instead. The result from Equation 5.1 is 5.93, which means that the student will still be able to graduate in a minimum of 6 semesters. While it may seem that this will facilitate little change in the schedule, the recommender system actually outputs a path that has a more balanced schedule. In the previous case, the credit hours per semester ranged between 13 and 18, but in this case all of the semester have 16

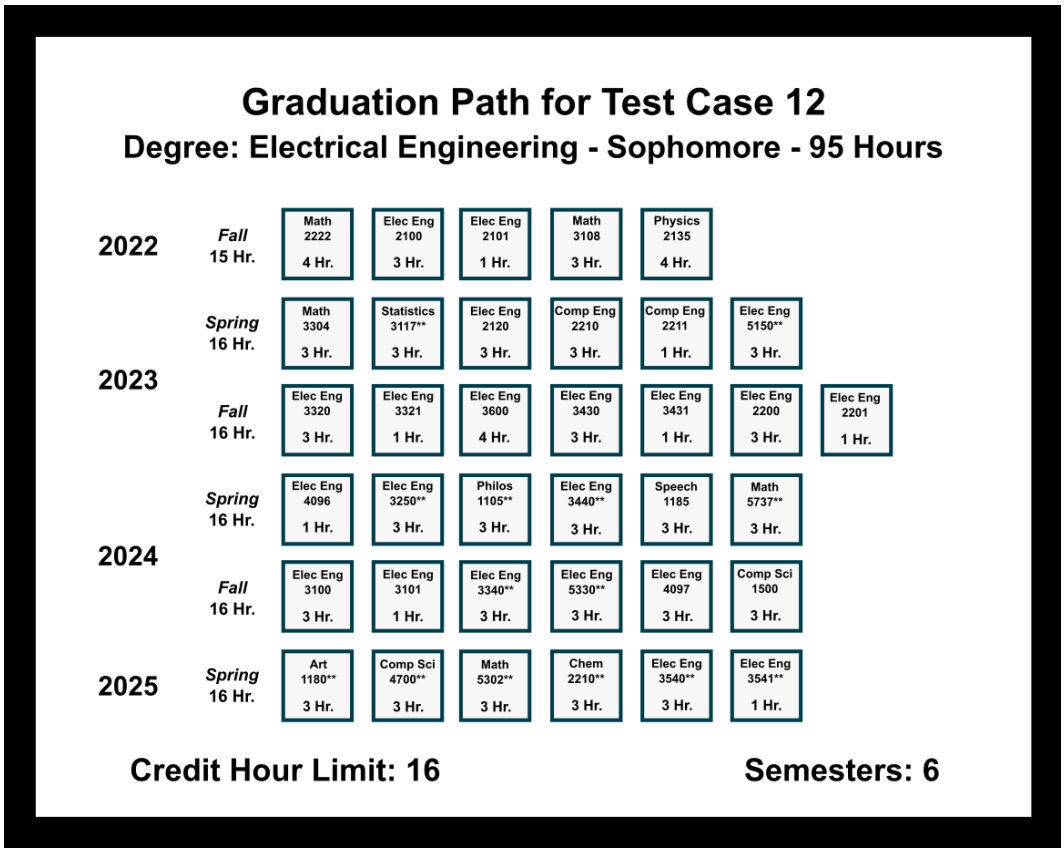


**denotes elective courses. All other courses are explicitly required.

Figure 5.11. Recommended graduation path for the test case 11.

hours except one, which has 15 hours. This balanced path could be preferrable for some, especially if they prefer a balanced course load and roughly equivalent levels of difficulty from semester to semester. Of course, this does not factor in the difficulty of the individual courses, but that could prove to be a fruitful area to explore in the future.

5.3.13. Test Case 13. The final test case (Figure 5.13) uses a student who is an incoming junior. This student has completed two years of courses, which are enumerated in Table 5.2. The student has set their credit hour limit to 18 hours and needs to completed 63 more hours in order to complete their degree. Using Equation 5.1, this means that the student has a minimum of 4 semesters remaining. The recommender system produces a schedule that perfectly matches those requirements and has the student graduating in 4 semesters. This case further shows that any student at any



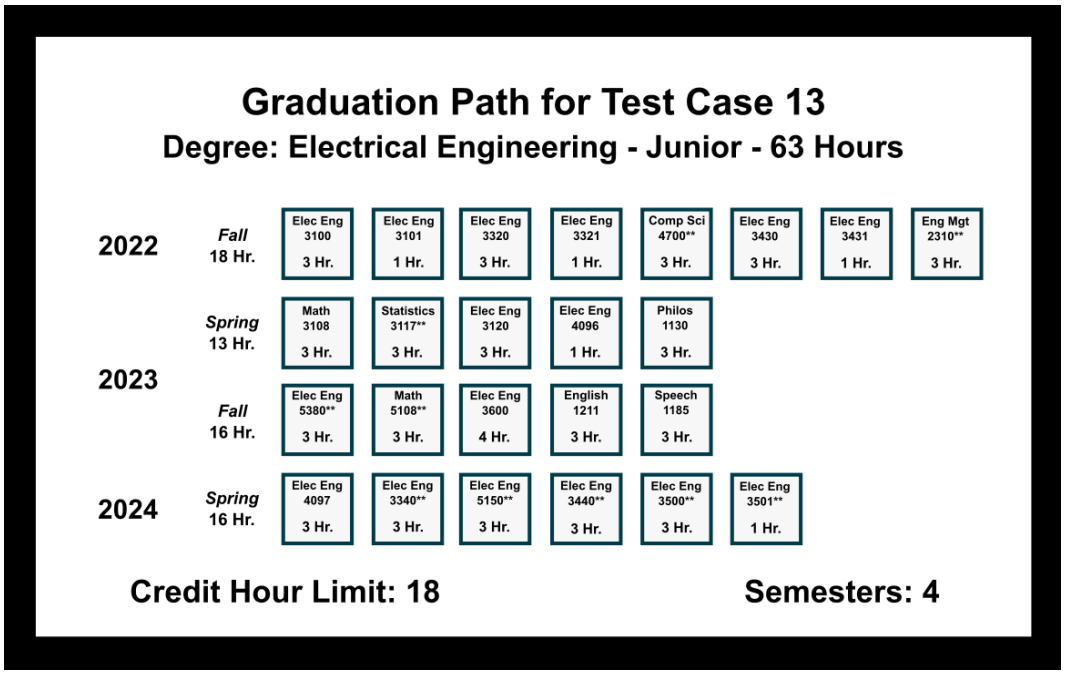
**denotes elective courses. All other courses are explicitly required.

Figure 5.12. Recommended graduation path for the test case 12.

point within their degree can use the recommender system and get meaningful results. The system could be very beneficial to junior and senior students who are looking for course suggestions and are able to provide keywords related to their interests.

5.4. SUMMARY

The recommender system proved to be very proficient at producing optimal graduation paths for a range of different students, credit hour limits, and even different interests. The system provides personalization based on individual interests and desired credit hour limits. The system returned optimal paths in each of the cases and followed all prerequisites and degree requirements, meaning that it could prove to be a valuable tool for universities when given a full dataset. Finally, the system does well at minimizing time-to-degree, which benefit students tremendously. Looking past the desired outcomes of the system, the test results show that the system offers potentially



**denotes elective courses. All other courses are explicitly required.

Figure 5.13. Recommended graduation path for the test case 13.

beneficial insight to school administrators as well. For example, the general chemistry course (Chem 1310, 1319, and 1100), which has no prerequisites, is a required course for in all three programs, yet it can be placed in practically any semester. The third test case has the course in the seventh semester, while the fifth test case has the course in the first semester. This means that despite the course being a basic required course, no other courses in the program require it. It could be argued that the course has little value to the student's degree and should not be a required course. While the system is mainly meant to help students generate graduation paths, running the system over various cases could also help highlight courses that should not be a part of specific programs. Overall, the results of the recommender system and its functionality is quite impressive.

6. CONCLUSION AND FUTURE DIRECTIONS

This thesis presented the course recommender system designed to fit into the PERCEPOLIS platform, starting with the literature review in Section 2. The literature review gave the reader a foundation of existing work in the field and and briefly discussed previous optimization work done within the PERCEPOLIS platform. Section 3 discussed the PERCEPOLIS platform in its current state and how the course recommender system fits into the larger project. The foundation established by Sections 2 and 3 set the stage for Section 4, which details the recommender system and the optimization problem within. With the components of the recommender system defined, Section 5 details the methods for testing the system and ensuring its proper functionality. Section 5 also discussed a number of test cases that were used on the system and their resulting graduation paths.

Overall, the recommender system's performance and resulting output are very promising, implementation of the system on a large scale gives it the potential to greatly reduce transfer and dropout rates by allowing students to concretely plan their degree and determine their own pacing. The fact that the system helps students complete their degree as quickly as possible is extremely valuable to universities, especially as students become increasingly wary of tuition, fees, and housing costs. At Missouri S&T, only 22% of students who started their degree in 2011 graduated within four years. That number was 23% for students who started in 2012. Meanwhile, about 63% graduated within six years for both sets of students [35]. It could be interpreted that, among other reasons, a majority of students at Missouri S&T are either taking unneeded courses, attending internships, or having to retake courses. The recommender system removes these unneeded courses and the credit hour limit, set by the student, allows the student to set their own pacing through their degree, which can prevent potential failure and the need to retake courses. Overall, the benefits that the system will have for students are extremely valuable and could save students thousands of dollars in tuition, housing, and fees by reducing the length of their college career and minimizing the number of hours that they take. Students that graduate earlier will enter the workforce earlier, positively affecting the national economy and potentially filling high-demand job openings, which is especially important in the current labor shortage. On a broader scale, the system has the potential to help thousands of students stay on track and focused through their degree, saving them valuable time and money in the long run.

There are several avenues for future contributions that build upon this work. In its current state, the recommender system requires the manual entry of course, degree, and student information. Entry of this information is quite burdensome and the required information is already stored in campus administration systems like PeopleSoft Campus Solutions [36]. Creation of an interface that allows the recommender system to import necessary data would prove to be a crucial time saver and would automate a majority of the repetitive work involved in using the recommender system. While interfacing to extract course and degree information is relatively straightforward, some universities may be hesitant to allow the extraction of student transcripts due to regulations put in place by the Family Educational Rights and Privacy Act (FERPA). Student transcripts are heavily guarded, and the student's permission is likely to be required before extraction, if extraction is permitted at all. With that in mind, creation of a transcript parser could prove to be a viable alternative. Although this would require the student to submit their transcript the extraction of data would remain automated, reducing the use of the recommender system to a short, minutes-long activity.

The recommender system was also developed specifically for use on bachelor's degrees, but was designed in a way that it can easily be expanded to include minors, master's degrees, and fast-track programs that allow students to combined bachelor's and master's degrees. This expansion would likely be valuable to prospective universities looking to use the system, as universities generally offer more than just bachelor's degrees. The expansion largely lies in the development of additional student types to account for differing rules for undergraduate and graduate students. For example, graduate students typically have a lower credit hour threshold to meet full-time status than undergraduate students do. Graduate students also have different degree requirements, often much broader, that could require new requirement formats to be developed. This expansion could be especially beneficial to students in fast-track programs, who would be able to plan out all of their degrees, from the beginning of the first one to the end of the last one.

A final extension to this work involves development of a machine learning system that uses a student's transcript to create a clustering profile for the student. This profile could then be used to match the student with a cluster of similar students, which could be used to suggest a credit hour limit based upon that of students that have graduated. The system could be designed in a way that allows it to review its previous suggestions, allowing it to make adjustments to its clustering and suggestion strategy in an attempt to improve its suggestions. As with many machine learning

systems, this method would require a substantial amount of training data to develop a basis for making recommendations. Development of such a system would be quite involved, but would prevent students from setting unrealistic credit hour limits.

APPENDIX

TEMPLATE GRADUATION PATHS

These are the template graduation paths that can be found in the Missouri S&T course catalog. There is one for each degree in the test data: Computer Science, Computer Engineering, and Electrical Engineering.

Freshman Year			
First Semester	**Credits**	**Second Semester**	**Credits**
FR ENG 1100	1	COMP SCI 1200	3
COMP SCI 1500[1]	3	COMP SCI 1570	3
Laboratory Science Elective[2]	5	COMP SCI 1580	1
MATH 1214[3]	4	MATH 1215[4]	4
ENGLISH 1120	3	ENGLISH 1160 or 3560	3
		Humanities / Social Science Elective[5]	3
	16		**17**
Sophomore Year			
First Semester	**Credits**	**Second Semester**	**Credits**
COMP SCI 1575	3	COMP SCI 2200	3
COMP SCI 1585	1	COMP SCI 2500	3
COMP ENG 2210[6]	3	PHYSICS 2135[9]	4
PHYSICS 1135[7]	4	COMP ENG 3150[6]	3
Statistics Elective[8]	3	Literature Elective[10]	3
Humanities / Social Science Elective[5]	3		
	17		**16**
Junior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
COMP SCI 2300	3	COMP SCI 3500	3
COMP SCI 3800	3	COMP SCI 3610	3
MATH 3108	3	Cmp Sc Elective[12, 16]	3
Humanities / Social Science Elective[5]	3	Sci/Eng Elective[13]	3
Ethics Elective[11]	3	SP&M S 1185[14]	3
	15		**15**
Senior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
COMP SCI 4090	3	COMP SCI 4091	3
COMP SCI 4610	3	Cmp Sc Electives[12, 16]	3
Cmp Sc Electives[12, 16]	6	Humanities / Social Science Elective[5]	3
Sci/Eng Elective[13]	3	Free Elective[15,16]	8
	15		**17**
Total Credits: 128			

Figure 1. Degree requirements for a B.S. degree in computer science at Missouri S&T.

Freshman Year			
First Semester	**Credits**	**Second Semester**	**Credits**
FR ENG 1100[2]	1	COMP SCI 1500	3
MATH 1214[3]	4	MATH 1215[3]	4
CHEM 1310	4	PHYSICS 1135[3,4]	4
CHEM 1319	1	ECON 1100 or 1200	3
HISTORY 1200, or 1300, or 1310, or POL SCI 1200	3	Elective-Hum or Soc (any level)[5]	3
ENGLISH 1120	3		
	16		**17**
Sophomore Year			
First Semester	**Credits**	**Second Semester**	**Credits**
ELEC ENG 2100[3,6,7]	3	COMP ENG 2210[3,6,8]	3
ELEC ENG 2101[3,6]	1	COMP ENG 2211[3,6]	1
MATH 2222[3]	4	ELEC ENG 2120[3,7,9]	3
COMP SCI 1570[3]	3	MATH 3304[3]	3
COMP SCI 1580[3]	1	COMP SCI 1200[3]	3
PHYSICS 2135[3,4]	4	COMP SCI 1575	3
	16		**16**
Junior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
COMP ENG 3110	3	COMP ENG Elective A[3,14]	3
COMP ENG 3150	3	ELEC ENG 3410[3,6,9]	3
COMP ENG 3151[3,6,8]	1	COMP SCI 3800 or 2500[3]	3
ELEC ENG 2200[3,6,7]	3	STAT 3117[12]	3
ELEC ENG 2201[3,6,7]	1	Communication Elective [13]	3
Mathematics Elective[10]	3		
SP&M S 1185[13]	3		
	17		**15**
Senior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
COMP ENG 5410[3]	3	COMP ENG Elective D[3,15,16]	3
COMP ENG Elective C[3,15,16]	3	COMP ENG Elective E[3,15,16]	3
COMP ENG 4096[3,17]	1	COMP ENG 4097[3,17]	3
Elective-Hum or Soc (any level)[5]	3	Professional Development Elective[20]	3
Engineering Science Elective [11]	3	Free Elective[18]	3
COMP ENG Elective B[3,19]	3		
	16		**15**
Total Credits: 128			

Figure 2. Degree requirements for a B.S. degree in computer engineering at Missouri S&T.

Freshman Year			
First Semester	**Credits**	**Second Semester**	**Credits**
FR ENG 1100[2]	1	MECH ENG 1720	3
CHEM 1310	4	MATH 1215[3]	4
CHEM 1319	1	PHYSICS 1135[3,4]	4
MATH 1214[3]	4	ECON 1100 or 1200	3
HISTORY 1200, or 1300, or 1310, or POL SCI 1200	3	Elective-Hum or Soc Sci (any level)[5]	3
ENGLISH 1120	3		
	16		**17**
Sophomore Year			
First Semester	**Credits**	**Second Semester**	**Credits**
ELEC ENG 2100[3,6,7]	3	ELEC ENG 2200[3,6,7,10]	3
ELEC ENG 2101[3,6]	1	ELEC ENG 2201[3,6,7]	1
MATH 2222[3]	4	ELEC ENG 2120[3,7,9]	3
COMP ENG 2210[3,6,8]	3	MATH 3304[3]	3
COMP ENG 2211[3,6]	1	Engineering Science Elective[11]	3
PHYSICS 2135[3,4]	4	COMP SCI 1500	3
	16		**16**
Junior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
ELEC ENG 3100[3,6,9,10]	3	ELEC ENG 3600[3,9]	4
ELEC ENG 3101[3,6,9,10]	1	El Eng Elective A[10,14,19]	3
ELEC ENG 3320	3	ELEC ENG 3430	3
ELEC ENG 3321	1	ELEC ENG 3431	1
SP&M S 1185[13]	3	STAT 3117[12]	3
MATH 3108	3	Communication Elective [13]	3
	14		**17**
Senior Year			
First Semester	**Credits**	**Second Semester**	**Credits**
El Eng Power Elective[3,6,9,15]	3	El Eng Elective C[10,14]	3
El Eng Power Elective Lab[3,6,9,15]	1	El Eng Elective E[17,19]	3
El Eng Elective B[10,14]	3	ELEC ENG 4097	3
El Eng Elective D[10,16,19]	3	Professional Development Elective[20]	3
ELEC ENG 4096[3]	1	Free Elective[18]	3
Free Elective[18]	3		
Elective-Hum or Soc Sci (any level)[5]	3		
	17		**15**
Total Credits: 128			

Figure 3. Degree requirements for a B.S. degree in electrical engineering at Missouri S&T.

REFERENCES

[1] of Engineering, N. A., 'NAE Grand Challenges for Engineering,' http://www.engineeringchallenges.org/challenges.aspx, 2022, accessed: 03-12-2022.

[2] 'Khan academy,' https://www.khanacademy.org/, 2022, accessed: 02-26-2022.

[3] 'Coursera,' https://www.coursera.org/, 2022, accessed: 02-26-2022.

[4] 'NCES full-time retention rate in postsecondary institutions,' https://nces.ed.gov/ipeds/TrendGenerator/app/build-table/7/32?f=5%3D1&rid=1&cid=2, 2020, accessed: 03-09-2022.

[5] 'NCES graduation within % of normal time (2012 cohort),' https://nces.ed.gov/ipeds/Search?query=graduation+within+200%25&query2=graduation+within+200%25&resultType=all&page=1&sortBy=relevance&overlayTableId=30450, 2020, accessed: 02-26-2022.

[6] Bernacki, M. L., Greene, M. J., and Lobczowski, N. G., 'A systematic review of research on personalized learning: Personalized by whom, to what, how, and for what purpose(s)?' Educational Psychology Review, December 2021, **33**(4), pp. 1675–1715, doi:10.1007/s10648-021-09615-8.

[7] Patrick, S., Kennedy, K., and Powell, A., 'Mean what you say: Defining and integrating personalized, blended and competency education,' October 2013.

[8] Ma, H., Wang, X., Hou, J., and Lu, Y., 'Course recommendation based on semantic similarity analysis,' in '2017 IEEE International Conference on Control Science and Systems Engineering (ICCSSE),' 2017 pp. 638–641, doi:10.1109/CCSSE.2017.8088011.

[9] Premalatha, M. and Viswanathan, V., 'Course sequence recommendation with course difficulty index using subset sum approximation algorithms,' Cybernetics and Information Technologies, September 2019, **19**, pp. 25–44, doi:10.2478/cait-2019-0024.

[10] Hu, Q., Polyzou, A., Karypis, G., and Rangwala, H., 'Enriching course-specific regression models with content features for grade prediction,' in '2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA),' 2017 pp. 504–513, doi:10.1109/DSAA.2017.74.

[11] Srifi, M., Hammou, B. A., Lahcen, A. A., and Mouline, S., 'A concise survey on content recommendations,' in '2018 International Conference on Big Data, Cloud and Applications,' 2018 pp. 393–405, doi:10.1007/978-3-319-96292-4_31.

[12] Klašnja-Milićević, A., Vesin, B., Ivanović, M., Budimac, Z., and Jain, L. C., 'Recommender systems in e-learning environments,' in 'E-Learning Systems: Intelligent Techniques for Personalization,' Number 112 in Intelligent Systems Reference Library, pp. 51–75, Springer International Publishing, 2017, doi:10.1007/978-3-319-41163-7_6.

[13] Yu, X., Wei, D., Chu, Q., and Wang, H., 'The personalized recommendation algorithms in educational application,' in '2018 IEEE International Conference on Information Technology in Medicine and Education (ITME),' 2018 pp. 664–668, doi:10.1109/ITME.2018.00153.

[14] Urdaneta-Ponte, M. C., Mendez-Zorrilla, A., and Oleagordia-Ruiz, I., 'Recommendation systems for education: Systematic review,' MDPI Electronics, July 2021, **10**, pp. 1–21, doi:10.3390/electronics10141611.

[15] Borges, G. and Stiubiener, I., 'Recommending learning objects based on utility and learning style,' in '2014 IEEE Frontiers in Education Conference (FIE) Proceedings,' 2014 pp. 1–9, doi:10.1109/FIE.2014.7044245.

[16] Jetinai, K., 'Rule-based reasoning for resource recommendation in personalized e-learning,' in '2018 IEEE International Conference on Information and Computer Technologies (ICICT),' 2018 pp. 150–154, doi:10.1109/INFOCT.2018.8356859.

[17] Zhu, Y., Mu, J., Tang, Q., Wang, J., Li, H., Li, Z., Wang, X., Yang, H., and Gao, Y., 'Application of intelligent course selection recommendation system based on ipv6,' in '2018 IEEE International Conference on Software Engineering and Service Science (ICSESS),' 2018 pp. 1–5, doi:10.1109/ICSESS.2018.8663842.

[18] He, H., Zhu, Z., Guo, Q., and Huang, X., 'A personalized e-learning services recommendation algorithm based on user learning ability,' in '2019 IEEE International Conference on Advanced Learning Technologies (ICALT),' 2019 pp. 318–320, doi:10.1109/ICALT.2019.00099.

[19] Bhumichitr, K., Channarukul, S., Saejiem, N., Jiamthapthaksin, R., and Nongpong, K., 'Recommender systems for university elective course recommendation,' in '2017 International Joint Conference on Computer Science and Software Engineering (JCSSE),' 2017 pp. 1–5, doi:10.1109/JCSSE.2017.8025933.

[20] Ganeshan, K. and Li, X., 'An intelligent student advising system using collaborative filtering,' in '2015 IEEE Frontiers in Education Conference (FIE),' 2015 pp. 1–8, doi:10.1109/FIE.2015.7344381.

[21] Morrow, T., Hurson, A. R., and Sedigh Sarvestani, S., 'A multi-stage approach to personalized course selection and scheduling,' in '2017 IEEE International Conference on Information Reuse and Integration (IRI),' 2017 pp. 253–262, doi:10.1109/IRI.2017.58.

[22] Rivera, J., 'A hybrid recommender system to enrollment for elective subjects in engineering students using classifcation algorithms,' International Journal of Advanced Computer Science and Applications, 2020, **11**, pp. 25–44, doi:10.2478/cait-2019-0024.

[23] Heileman, G., Abdallah, C. T., Slim, A., and Hickman, M., 'Curricular analytics: A framework for quantifying the impact of curricular reforms and pedagogical innovations,' 2018, pp. 1–29.

[24] Xu, J., Moon, K. H., and van der Schaar, M., 'A machine learning approach for tracking and predicting student performance in degree programs,' IEEE Journal of Selected Topics in Signal Processing, April 2017, **11**(5), pp. 742–753, doi:10.1109/JSTSP.2017.2692560.

[25] Hellas, A., Ihantola, P., Petersen, A., Ajanovski, V. V., Gutica, M., Hynninen, T., Knutas, A., Leinonen, J., Messom, C., and Liao, S. N., 'Predicting academic performance: A systematic literature review,' in '23rd Annual ACM Conference on Innovation and Technology in Computer Science Education,' 2018 p. 175–199, doi:10.1145/3293881.3295783.

[26] Aguiar, E., Chawla, N. V., Brockman, J., Ambrose, G. A., and Goodrich, V., 'Engagement vs performance: Using electronic portfolios to predict first semester engineering student retention,' in 'Proceedings of the Fourth International Conference on Learning Analytics And Knowledge,' 2014 p. 103–112, doi:10.1145/2567574.2567583.

[27] 'Figma,' https://www.figma.com/, 2022, accessed: 04-18-2022.

[28] 'Angular,' https://angular.io/, 2022, accessed: 04-18-2022.

[29] 'Typescript,' https://www.typescriptlang.org/, 2022, accessed: 04-18-2022.

[30] 'Spring boot,' https://spring.io/projects/spring-boot, 2022, accessed: 04-18-2022.

[31] 'Kotlin,' https://kotlinlang.org/, 2022, accessed: 04-18-2022.

[32] 'Postgresql,' https://www.postgresql.org/, 2022, accessed: 04-18-2022.

[33] 'Google OR tools,' https://developers.google.com/optimization/cp/cp_solver, 2022, accessed: 02-26-2022.

[34] 'Missouri S&T course catalog,' https://catalog.mst.edu/undergraduate/, 2022, accessed: 03-04-2022.

[35] 'NCES Missouri S&T graduation rates (2012 cohort),' https://nces.ed.gov/ipeds/datacenter/institutionprofile.aspx?unitId=178411, 2020, accessed: 02-26-2022.

[36] 'PeopleSoft Campus Solutions,' https://docs.oracle.com/cd/E52319_01/infoportal/cs.html, 2022, accessed: 02-26-2022.

VITA

Nicolas Charles Dobbins received his Bachelor of Science in Computer Engineering from the Missouri University of Science and Technology in May 2021. During his undergraduate career, he participated in undergraduate research with Dr. Sahra Sedigh Sarvestani and Dr. Alireza Hurson on the PERCEPOLIS project, which he continued into his master's program. He continued his education at the Missouri University of Science and Technology, receiving his Master of Science in Computer Engineering in July 2022. He worked as a Graduate Research Assistant from 2021 to 2022. In addition, he began working for the Ameren Corporation in 2019 as an intern, where he has been able to collaborate on several software development projects. In May 2022, he became a full-time employee, working as a Network Engineer doing development, support, and maintenance for several electrical analysis-based applications.