01 Jan 2023

Personalizing Student Graduation Paths Using Expressed Student Interests

Nicolas Dobbins

Ali R. Hurson
Missouri University of Science and Technology, hurson@mst.edu

Sahra Sedigh
Missouri University of Science and Technology, sedighs@mst.edu

Recommended Citation

N. Dobbins et al., "Personalizing Student Graduation Paths Using Expressed Student Interests," *Proceedings - International Computer Software and Applications Conference*, pp. 142 - 151, Institute of Electrical and Electronics Engineers, Jan 2023.
The definitive version is available at https://doi.org/10.1109/COMPSAC57700.2023.00027

2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC) | 979-8-3503-2697-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/COMPSAC57700.2023.00027

Personalizing Student Graduation Paths Using Expressed Student Interests

Nicolas Dobbins, Ali R. Hurson, and Sahra Sedigh Sarvestani
Department of Electrical and Computer Engineering
Missouri University of Science and Technology
Rolla, MO, USA
{ncdz3b, hurson, sedighs}@mst.edu

***Abstract*—This paper proposes an intelligent recommendation approach to facilitate personalized education and help students in planning their path to graduation. The goal is to identify a path that aligns with a student's interests and career goals and approaches optimality with respect to one or more criteria, such as time-to-graduation or credit hours taken. The approach is illustrated and verified through application to undergraduate curricula at the Missouri University of Science and Technology.**

***Index Terms*—recommendation, optimization, personalized education, PERCEPOLIS**

I. Introduction

Personalized learning was identified in 2008 by the National Academy of Engineering as one of fourteen grand challenges for engineering for the 21st century [1]. Educational technology and related research has risen to the challenge leading to different implementations of personalized learning, most often in online learning systems and workplace learning management systems. Despite this progress, large-scale deployment at the university level remains limited to customization of course content on learning management systems.

Educational technology is especially sparse in solutions that facilitate multi-year planning by students and advisors, who often select (or recommend) courses one semester in advance. Very few students begin their college career with knowledge of the specific courses they will take for each semester, because drawing out a full path to graduation is a tedious and time-consuming task. Even those who do take the time to plan out a full path to graduation are not able to exhaustively consider the hundreds, if not thousands of possible schedules. As a result, many students select courses solely to satisfy their degree requirements without considering content that aligns with their interests and career goals. Others may end up with disproportionately heavy course loads due to lack of advance planning, leading them to frustration, a drop in performance, or delay in graduation.

The original research described in this paper aims to address these shortcomings by creating an intelligent recommender system that personalizes and optimizes the path to graduation with consideration of curricular requirements and the student's academic history, interests, and future career goals. The proposed approach provides a student with a semester-by-semester schedule of recommended courses by formulating the selection of courses as a multi-objective optimization problem. These objectives 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 approach is illustrated through several examples that are based on actual curricula at the Missouri University of Science and Technology (S&T).

The proposed approach is intelligent and dynamic and enables advance planning that can lead to more balanced course loads, a more cohesive plan of study, greater engagement due to alignment of courses with interests, and shorter time-to-graduation, which typically correlates with lower costs.

II. Related Work

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 [2], but [3] 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: recommendation [4], path optimization [5], and prediction [6].

The *recommender* 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, and workforce training systems. In contrast, the *path optimization* approach focuses on optimizing a student's graduation path based upon prerequisites and a balanced course load. Studies in this area serve as the basis for 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.

The approach proposed in this paper is unique in pairing the recommendation approach with path optimization.

The authors gratefully acknowledge funding from NSF DUE-1742523 and the Intelligent Systems Center at the Missouri University of Science and Technology.

979-8-3503-2697-0/23/$31.00 ©2023 IEEE
DOI 10.1109/COMPSAC57700.2023.00027

A. Recommender Systems

Recommendation algorithms have become increasingly prevalent over the past few years due to the growing expanse of data available to and about consumers. These algorithms are designed to make meaningful recommendations to a consumer based on their profile and the profiles of users identified as similar to them. There are two main approaches to content recommendation: content-based and collaborative-filtering.

The *content-based* approach is generally used for recommendation algorithms within streaming applications and other similar content-providing applications [7]. This method uses the user's content history to identify similar content. The approach assumes that most users have a specific type of content that interests them and relies on that assumption to provide useful recommendations. An example of this method is a course recommender system, where the student inputs a course that they know about and in return receives a list of similar courses [4]. Another example is the system created by Borges and Stiubiener, which uses both user interests and learning ability as a means to suggest learning resources to the user [8].

On the other hand, the *collaborative-filtering* approach relies on knowledge of a user's preferences or characteristics and aims to identify content that has been perused by similar users. In this approach, users are clustered into classes of similarity based on different parameters [9]. Provided that the classification of the user is correct, the resulting set of recommendations generally tends to be more personalized.

Content-based filtering relies on knowledge of a user's history and considers relationships between different items of content. This information may not be available in all scenarios, leading to the greater prevalence of collaborative filtering [10].

Several examples of recommendation algorithms in learning technology are enumerated in [11] and [12], respectively. A notable example is an educational recommender system that considers user information and behavior to calculate similarity between users [13]. The system then uses the user's learning ability to further refine the measure of similarity, allowing the system to recommend e-learning services based upon the user's similarity to a specific set of previous users. Another system clusters students based upon their course history [14]. This allows the system to identify a similar user or set of similar users who are further along in their program and makes suggestions based on courses taken by the similar user(s). The system proposed by Ganeshan and Li creates clusters based on basic student attributes, including GPA, age, ethnicity, and gender [15] and makes course recommendations based on the history of users in the cluster.

B. Path Optimization Approaches

One approach to course recommendation is *path optimization*, i.e., visualizing the curriculum as a graph and applying graph-theoretical methods to identify the "best" route between source (entering the program) and destination (graduation).

Notable among related studies is [5], which focuses on optimizing graduation paths while providing a difficulty rating for each course in the path. The approach is largely based on PERCEPOLIS [16], which utilizes prerequisites and a credit hour limit as constraints to the optimization problem. The approach in [5] assigns difficulty ratings to each of the courses in the graduation path, which are then used to help balance the student's semester load. Another system, Curricular Analytics, aims to identify bottlenecks and other potential areas of improvement within in a degree program [17].

The closest work to our proposed approach is [16], where a two-step method was used to populate 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 took the resulting set of courses and placed them into an optimized semester-by-semester ordering based on the number of courses for which each course serves as a prerequisite. This method prioritizes placement of "critical path" courses (which unlock numerous other courses) in order to shorten the student's time-to-degree. The work described in this paper builds on [16], but considerably improves on the earlier method by concurrently selecting courses and placing them into an optimal ordering. This allows for the best course, with respect to fulfilling program requirements during a specific semester and the overall path to graduation, to be recommended at any given point in the path.

C. Predictive Approaches

Predictive approaches focus on attempting to predict a student's probability of "success", which could refer to passing a given course or completing an entire academic program. The system proposed by Hu et. al. used a regression process as the basis for prediction [6]. Each course was rated based on difficulty, academic level, and quality of the instructor. The predictive model proposed by Xu et. al. 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 [18]. This approach takes the student's background and schedule 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 grade that the student will earn in each course, which in turn allows prediction of the semester GPA. Interested readers are referred to [19] for a comprehensive review of similar studies.

III. Methodology

Design of the recommendation algorithm was an involved process, due to the numerous constraints that need to be met to generate even a feasible (not necessarily optimal) schedule that would meet all program requirements. As depicted in Fig. 1, 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 keywords corresponding to their interests and career goals. The algorithm aims to select courses that align with these interests and goals to fill as many degree requirements as possible, maximizing the student's college career experience

TABLE I
NOMENCLATURE.

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 upper bound
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

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 where the solution is an "optimal" semester-by-semester schedule of courses for the remainder of the student's program of study. The optimization simultaneously selects and schedules courses into a near-optimal ordering so that the student can graduate as quickly as possible with their given pacing. Fig. 2 depicts the optimization problem, including its constraints, decision variables, and objective function, each of which is described in the remainder of this section. A list of nomenclature is provided in Table I.

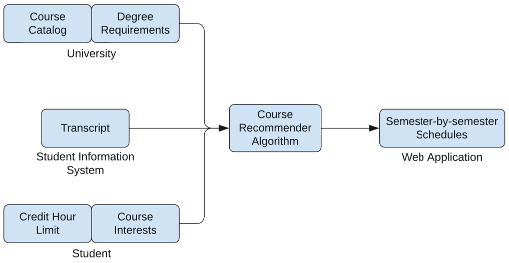


Fig. 1. Input and output information for the recommendation algorithm.

A. 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. (1) defines 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.

$$r_n = \{c : c \in S_n\} \, and \, r_n \subseteq S_n \tag{1}$$

B. 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 (2) ensures that unrepeatable courses are only taken once.

$$R = \bigcup_{n=1}^{N} r_n \, and \, \{c \in R\} = 1 \tag{2}$$

C. 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 (3) ensures that the credit hour limit is never surpassed in a given semester.

$$h_n = (\sum h_c, \forall c \in r_n) \leq CHL \tag{3}$$

In addition to the semester credit hour constraint, many degrees require that a number of hours be completed for graduation. (4) defines this parameter as a constraint for the optimization problem.

$$H_p + \sum_{n=1}^{N} h_n \geq CHR \tag{4}$$

D. 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 (5), as a part of the optimization process, they will be enforced by the optimizer.

$$degreeAudit(C \bigcup R) = True \tag{5}$$

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.

E. 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 ((6)).

$$P_n \subseteq \{\bigcup_{i=1}^{n-1} r_i\} \cup C \tag{6}$$

In short,(6) ensures that all prerequisites for the courses in a semester have been taken prior to the given semester.

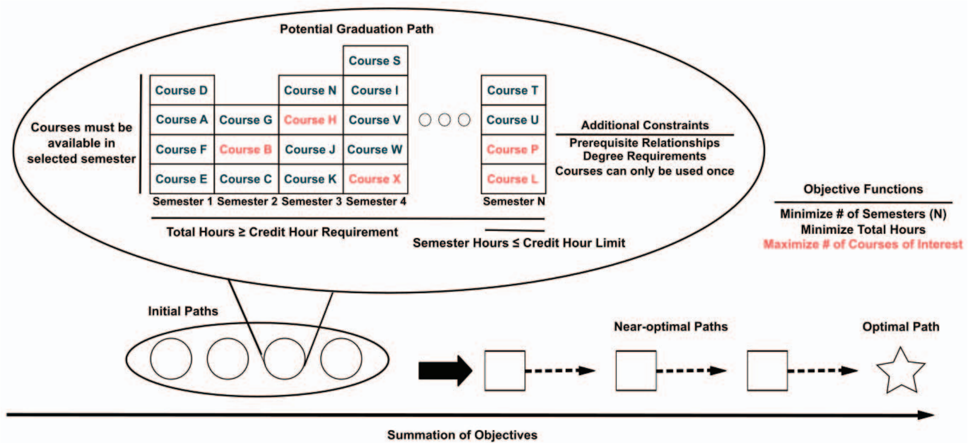


Fig. 2. Multi-criteria optimization of the path to graduation.

F. 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 (7).

$$Q_n \subseteq \{\bigcup_{i=1}^{n} r_i\} \cup C \tag{7}$$

G. 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 (8) and (9), respectively.

$$\text{Minimize } \{max(n) : r_n \neq \emptyset\} \tag{8}$$

$$\text{Minimize } \sum_{n=1}^{N} h_n \tag{9}$$

The secondary objective function is intended to achieve 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 (10).

$$\text{Maximize } R_{int} : R_{int} \subseteq R \tag{10}$$

The recommender system solves the multi-objective optimization problem using the Google OR Tools CP-SAT solver [20]. 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. CP-SAT 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 the 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, which are not necessarily optimal.

IV. Validation

The validation process included creating a database for three degree programs: computer engineering (CpE), computer science (CS), and electrical engineering (EE). The database contained all major courses, elective courses, pre/co-requisite relationships among courses, and degree requirements for each degree program. These degree requirements and relationships among courses were extracted from the S&T course catalog [21]. A student database was also created, with variations in academic history and interests among the test students. The proposed recommender system was run for each case and the recommended schedules were recorded and verified. These cases allowed for testing against both new (incoming) students and students who had already completed one or more semesters. The recommended schedule for each test case was checked for feasibility by checking that degree and pre/corequisite requirements were met. It was then checked for efficiency by determining whether a change would shorten the time to degree. The results are described in detail in the remainder of this section.

A. Test data

The test data used for testing and validating the recommender system included detailed requirements for three degree programs (CpE, CS, and EE) and all associated requirements and required courses, projected course offerings for several semesters (from 2022 up to 2030), and a wide range of courses. The courses included in the test data consist of all courses from CpE, CS, EE, and math , 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 II 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 S&T course catalog [21].

B. Test cases

Tables III and IV, respectively enumerate selected test cases and their respective parameters. These test cases are collectively designed to show the capabilities and flexibility of the proposed recommender system. Three of the 13 act as base cases; the remaining ten are variations. Each base case belongs to one of the selected degree programs, namely, CpE, CS, or EE. These base cases allow for validation of degree requirements, prerequisites, corequisites, and credit hour requirements. As such, these cases use a new (incoming) student who has received entry credit for the basic math courses (Math 1140 and Math 1160), a credit hour limit of 18, and no declared interest. This credit hour limit of 18 was chosen because the university 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 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 show 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 show that each of these individual cases can be combined and near-optimal results will be achieved.

TABLE II
A list of keywords and related courses.

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

TABLE III
Test cases 1 through 9.

Case	Program	CHL	Interests	Courses Taken
1	CS	18	-	-
2	CS	13	Gaming	-
3	CS	19	-	-
4	CS	16	-	-
5	CpE	18	-	-
6	CpE	15	Comp. Architecture	-
7	CpE	19	Comp. Architecture	-
8	EE	18	-	-
9	EE	19	-	-

C. Test Results

The analysis of the generated graduation path for each case proved the validity of the proposed system. The recommender

TABLE IV
TEST CASES 10 THROUGH 13.

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

system found an optimal solution for each case and successfully generated graduation paths that fulfill all pre/corequisite 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 identifies a near-optimal solution with respect to all three objectives: minimizing time-to-degree, minimizing credit hours, and maximizing courses of interest. The remainder of this section describes the results for selected test cases. The remainder have been omitted in the interest of brevity.

Please note that thick black edges between two courses denote prerequisite (vertical) or corequisite (horizontal) relationships. Only some relationships are shown; many more exist. Courses marked with asterisks (**) are elective. All other courses are explicitly required.

Test case 1, shown in Fig. 3, demonstrates the system's ability to output a valid graduation path for a new student in the CS program. This is a base case for CS program with an 18 credit hour limit. 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 seven semesters being needed for graduation, as compared to the typical eight-semester graduation time that results from a default program. A simple way to check for the minimum number of semesters required to graduate includes calculating (11). It is important to note that the student has received five 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. In this case, the path contains seven 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 declare any interests, so there was no opportunity to maximize the number of courses of interest in the schedule.

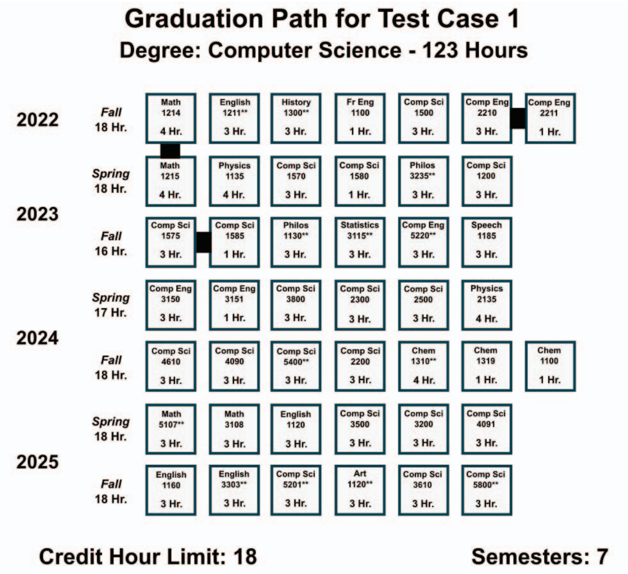
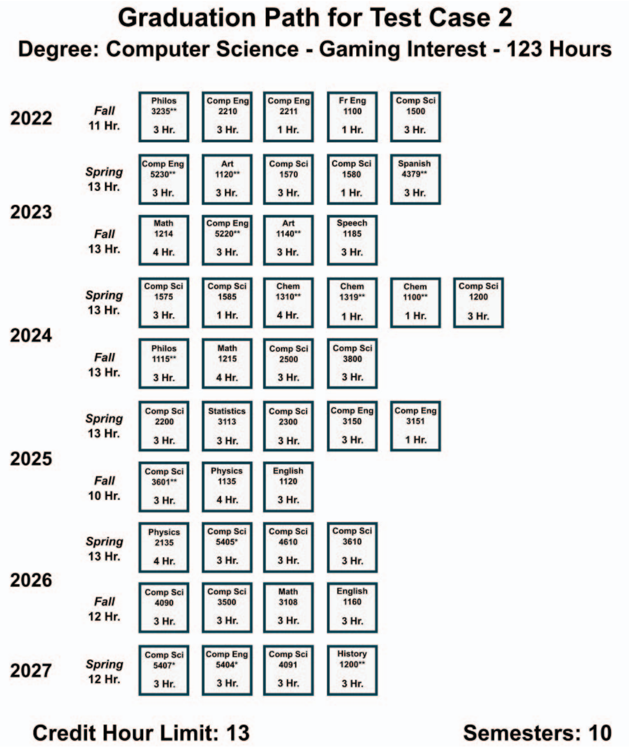


Fig. 3. Recommended path for test case 1.

$$\frac{CHR - \text{Hours completed}}{CHL} = \text{Min semesters} \qquad (11)$$

Test case 2 is similar to the first case, with the difference of expressing interest in "gaming" and using a 13-hour limit. As seen in Fig. 4, the resulting graduation path includes 10 semesters and the minimum of 123 credit hours. Since the credit hour limit changed, the result of (11) is 9.46, meaning that the system has minimized time-to-degree. 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 CS, this is the maximum number of courses that the system can use, due to the degree requirements. The CS program only allows for three 5000-level elective courses from CS. Even with this constraint, the system still maximized the number of courses of interest and provided an optimized graduation path.

Test case 3, shown in Fig. 5 is also similar to the first test case, except the credit hour limit has been increased



Graduation Path for Test Case 2

Degree: Computer Science - Gaming Interest - 123 Hours

2022	Fall 11 Hr.	Philos 3235** 3 Hr.	Comp Eng 2210 3 Hr.	Comp Eng 2211 1 Hr.	Fr Eng 1100 1 Hr.	Comp Sci 1500 3 Hr.	
2023	Spring 13 Hr.	Comp Eng 5230** 3 Hr.	Art 1120** 3 Hr.	Comp Sci 1570 3 Hr.	Comp Sci 1580 1 Hr.	Spanish 4379** 3 Hr.	
	Fall 13 Hr.	Math 1214 4 Hr.	Comp Eng 5220** 3 Hr.	Art 1140** 3 Hr.	Speech 1185 3 Hr.		
2024	Spring 13 Hr.	Comp Sci 1575 3 Hr.	Comp Sci 1585 1 Hr.	Chem 1310** 4 Hr.	Chem 1319** 1 Hr.	Chem 1100** 1 Hr.	Comp Sci 1200 3 Hr.
	Fall 13 Hr.	Philos 1115** 3 Hr.	Math 1215 4 Hr.	Comp Sci 2500 3 Hr.	Comp Sci 3800 3 Hr.		
2025	Spring 13 Hr.	Comp Sci 2200 3 Hr.	Statistics 3113 3 Hr.	Comp Sci 2300 3 Hr.	Comp Eng 3150 3 Hr.	Comp Eng 3151 1 Hr.	
	Fall 10 Hr.	Comp Sci 3601** 3 Hr.	Physics 1135 4 Hr.	English 1120 3 Hr.			
2026	Spring 13 Hr.	Physics 2135 4 Hr.	Comp Sci 5405* 3 Hr.	Comp Sci 4610 3 Hr.	Comp Sci 3610 3 Hr.		
	Fall 12 Hr.	Comp Sci 4090 3 Hr.	Comp Sci 3500 3 Hr.	Math 3108 3 Hr.	English 1160 3 Hr.		
2027	Spring 12 Hr.	Comp Sci 5407* 3 Hr.	Comp Eng 5404* 3 Hr.	Comp Sci 4091 3 Hr.	History 1200** 3 Hr.		

Credit Hour Limit: 13 Semesters: 10

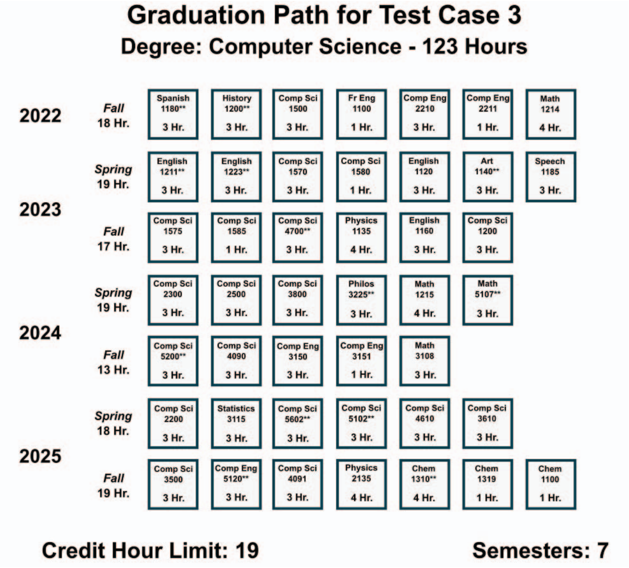
Fig. 4. Recommended path for test case 2.

to 19 hours. Due to this change, the result of (11) is now 6.47, meaning that despite increasing the limit, the minimum time-to-degree is still seven semesters. The recommended path contains seven semesters and 123 hours, both of which match their respective minima. The biggest difference between the first case and this case is the variation in the number of hours in each semester, which range from 13 to 19. Increasing the credit hour limit does not reduce the time-to-degree, but 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 preparation and planning.

D. *Test Case 3*

Test case 4, decreases the credit hour limit to 16. This means that the result of (11) becomes 7.68, and this student will graduate in a minimum of eight semesters. As is shown in Fig. 6, the recommender system outputs a path with eight semesters and the minimum 123 hours.

Test case 5, serves as the second base case, this time for the CpE program. 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 (11) on this case is 7.05, implying a minimum of eight semesters to graduation. The result of the recommender system, shown in



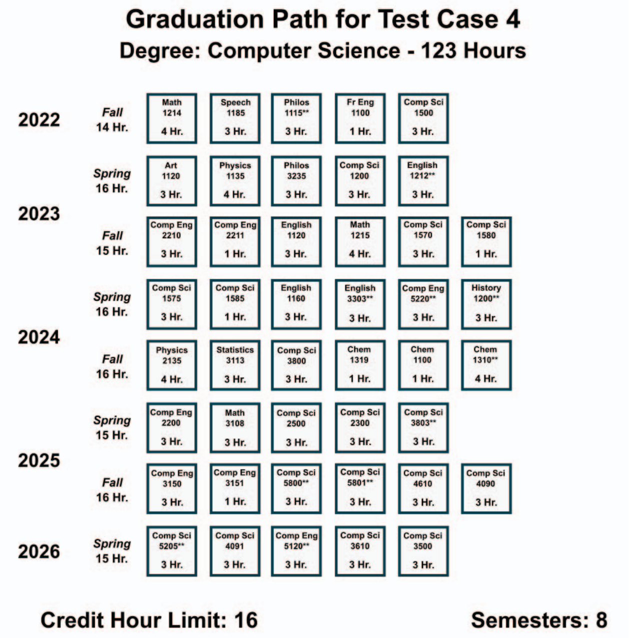
Graduation Path for Test Case 3

Degree: Computer Science - 123 Hours

2022	Fall 18 Hr.	Spanish 1180** 3 Hr.	History 1200** 3 Hr.	Comp Sci 1500 3 Hr.	Fr Eng 1100 1 Hr.	Comp Eng 2210 3 Hr.	Comp Eng 2211 1 Hr.	Math 1214 4 Hr.
2023	Spring 19 Hr.	English 1211** 3 Hr.	English 1223** 3 Hr.	Comp Sci 1570 3 Hr.	Comp Sci 1580 1 Hr.	English 1120 3 Hr.	Art 1140** 3 Hr.	Speech 1185 3 Hr.
	Fall 17 Hr.	Comp Sci 1575 3 Hr.	Comp Sci 1585 1 Hr.	Comp Sci 4700** 3 Hr.	Physics 1135 4 Hr.	English 1160 3 Hr.	Comp Sci 1200 3 Hr.	
2024	Spring 19 Hr.	Comp Sci 2300 3 Hr.	Comp Sci 2500 3 Hr.	Comp Sci 3800 3 Hr.	Philos 3225** 3 Hr.	Math 1215 4 Hr.	Math 5107** 3 Hr.	
	Fall 13 Hr.	Comp Sci 5200** 3 Hr.	Comp Sci 4090 3 Hr.	Comp Eng 3150 3 Hr.	Comp Eng 3151 1 Hr.	Math 3108 3 Hr.		
2025	Spring 18 Hr.	Comp Sci 2200 3 Hr.	Statistics 3115 3 Hr.	Comp Sci 5602** 3 Hr.	Comp Sci 5102** 3 Hr.	Comp Sci 4610 3 Hr.	Comp Sci 3610 3 Hr.	
	Fall 19 Hr.	Comp Sci 3500 3 Hr.	Comp Eng 5120** 3 Hr.	Comp Sci 4091 3 Hr.	Physics 2135 4 Hr.	Chem 1310** 4 Hr.	Chem 1319 1 Hr.	Chem 1100 1 Hr.

Credit Hour Limit: 19 Semesters: 7

Fig. 5. Recommended path for test case 3.



Graduation Path for Test Case 4

Degree: Computer Science - 123 Hours

2022	Fall 14 Hr.	Math 1214 4 Hr.	Speech 1185 3 Hr.	Philos 1115** 3 Hr.	Fr Eng 1100 1 Hr.	Comp Sci 1500 3 Hr.	
2023	Spring 16 Hr.	Art 1120 3 Hr.	Physics 1135 4 Hr.	Philos 3235 3 Hr.	Comp Sci 1200 3 Hr.	English 1212** 3 Hr.	
	Fall 15 Hr.	Comp Eng 2210 3 Hr.	Comp Eng 2211 1 Hr.	English 1120 3 Hr.	Math 1215 4 Hr.	Comp Sci 1570 3 Hr.	Comp Sci 1580 1 Hr.
2024	Spring 16 Hr.	Comp Sci 1575 3 Hr.	Comp Sci 1585 1 Hr.	English 1160 3 Hr.	English 3303** 3 Hr.	Comp Eng 5220** 3 Hr.	History 1200** 3 Hr.
	Fall 16 Hr.	Physics 2135 4 Hr.	Statistics 3113 3 Hr.	Comp Sci 3800 3 Hr.	Chem 1319 1 Hr.	Chem 1100 1 Hr.	Chem 1310** 4 Hr.
2025	Spring 15 Hr.	Comp Eng 2200 3 Hr.	Math 3108 3 Hr.	Comp Sci 2500 3 Hr.	Comp Sci 2300 3 Hr.	Comp Sci 3803** 3 Hr.	
	Fall 16 Hr.	Comp Eng 3150 3 Hr.	Comp Eng 3151 1 Hr.	Comp Sci 5800** 3 Hr.	Comp Sci 5801** 3 Hr.	Comp Sci 4610 3 Hr.	Comp Sci 4090 3 Hr.
2026	Spring 15 Hr.	Comp Sci 5205** 3 Hr.	Comp Sci 4091 3 Hr.	Comp Eng 5120** 3 Hr.	Comp Sci 3610 3 Hr.	Comp Sci 3500 3 Hr.	

Credit Hour Limit: 16 Semesters: 8

Fig. 6. Recommended path for test case 4.

Fig. 7 is a path that achieves both minima: 8 semesters and 127 hours. The path suggested by the university for this program takes eight semesters and 128 credit hours. The recommender system saves one credit hour by considering the credit earned for the two basic math courses.

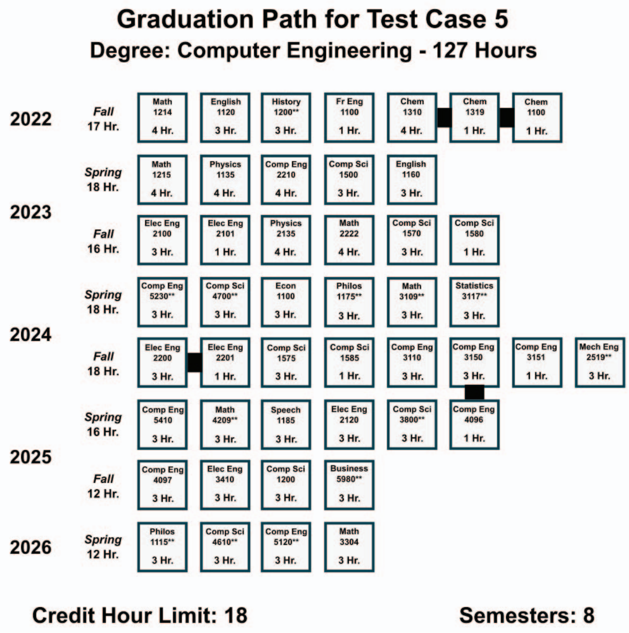


Fig. 7. Recommended path for test case 5.

Test Case 6 is the same as Test Case 5, but the student expresses an interest in computer architecture and sets the credit hour limit to 15 Due to the change in the credit hour limit, the minimum number of semesters is now nine, but the minimum number of hours needed is still 127. The recommender system outputs a path that achieves these minima, as shown in Fig. 8. In contrast to the previous case, the recommender system selects five computer architecture courses as electives. The courses of interest are: CpE 5110, 5120, 5130, 5151, and 5160. These five courses are the maximum number that can be placed into the path, because the other courses in the area of interest do not meet the 5000-level requirement of the elective. Since the recommender system prioritizes time-to-degree before courses of interest, it will not place unneeded courses into the path at the expense of adding additional semesters or hours.

Test Case 7 increases the credit hour limit to 19 hours. This means that the result of (11) is 6.68 and a minimum of seven semesters is needed. The recommended graduation path, shown on the right of Fig. 9(b) spans seven semesters and 127 hours. The path also includes the same five courses of interest as in the previous case. This path is optimal and 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 seen on the left of Fig. 9 is the graduation path for an actual student who has completed their degree. This student graduated in seven semesters and took 129 credit hours in total, with 22 credit hours from credit taken in high school. Looking at the student's CpE 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 CpE. Comparing to the test case, the selection of courses is arguably random. If the student had expressed an interest in networks or computer architecture and used the recommender system, their path would have been more cohesive.

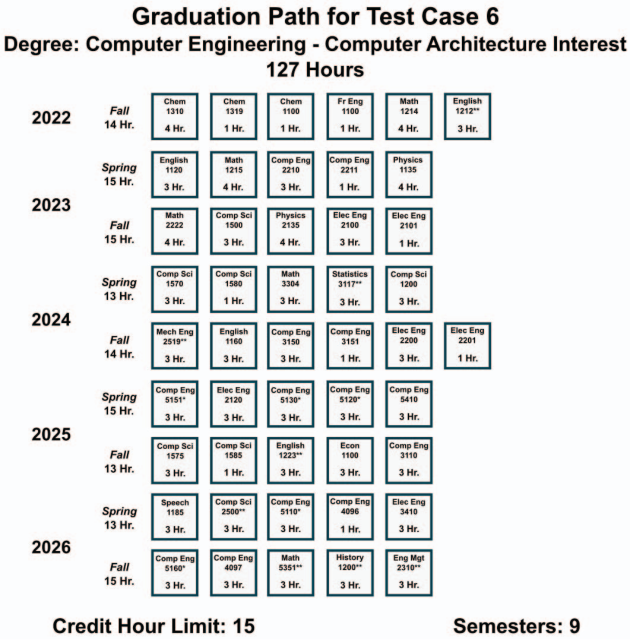


Fig. 8. Recommended path for test case 6.

Test Case 11 considers an incoming sophomore who completed a number of courses during their freshman year. The courses taken, shown in Table IV were chosen based upon the suggested freshman schedule in the example path for the EE program (omitted for brevity). The student needs 95 more credit hours to complete their degree. This case uses an 18 credit hour limit, which means the result of (11) is 5.28 and at least six more semesters are required. The path generated by the recommender system, shown in Fig. 10 achieves the computed minima.

Test Case 12 considers the same student as Test Case11, except the student specifies a 16 credit hour limit instead of 18. The result of (11) 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 recommended path (shown in Fig. 11) has a more balanced load. In the previous case, the credit hours per semester ranged between 13 and 18, but in this case each

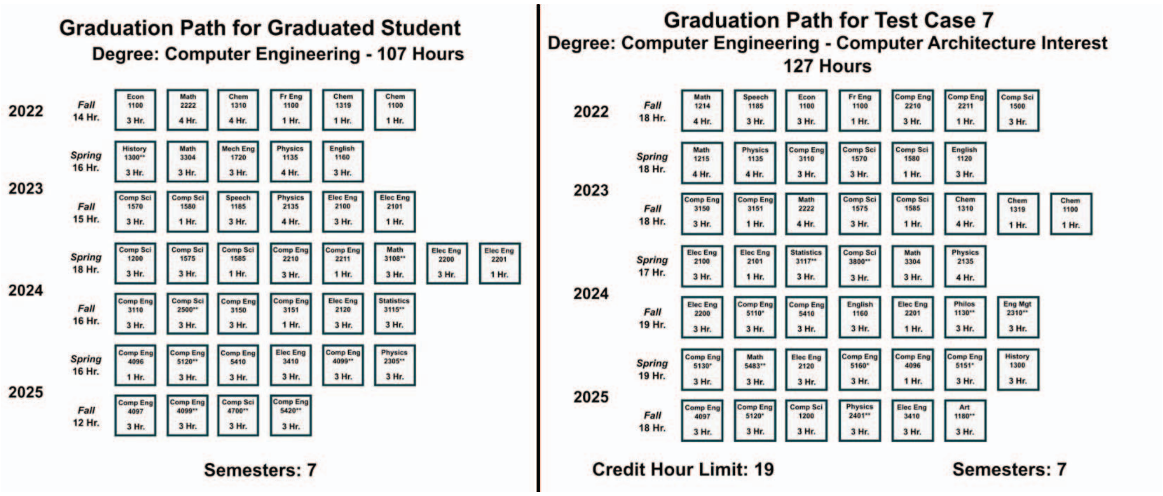


Fig. 9. (a) Actual path for a graduated student (left) and (b) the recommended graduation path for test case 7 (right).

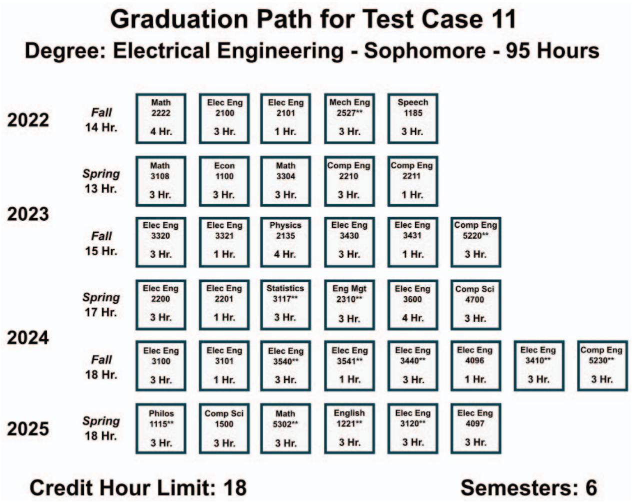


Fig. 10. Recommended path for test case 11.

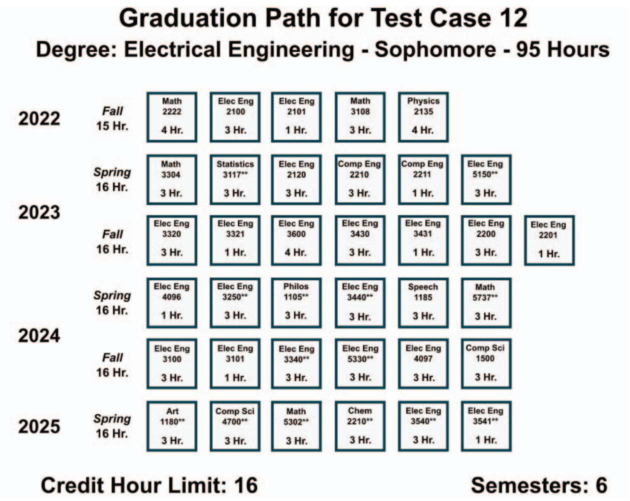


Fig. 11. Recommended path for test case 12.

semester has 16 hours except for one, which has 15. This balanced path could be preferable for some students.

The final test case, **Test Case 13**, considers an incoming junior who completed two years of courses, as enumerated in Table IV. The student has set their credit hour limit to 18 hours and needs to complete 63 more hours in order to complete their degree. Based on (11), a minimum of four semesters remain. The recommended schedule, shown in Fig. 12 achieves this minimum. The system could be very beneficial to junior and senior students who are looking for course suggestions, especially if they are able to provide keywords related to their interests.

The recommender system proved capable of producing optimal graduation paths for students with different academic histories, credit hour limits, and interests. Optimal paths that

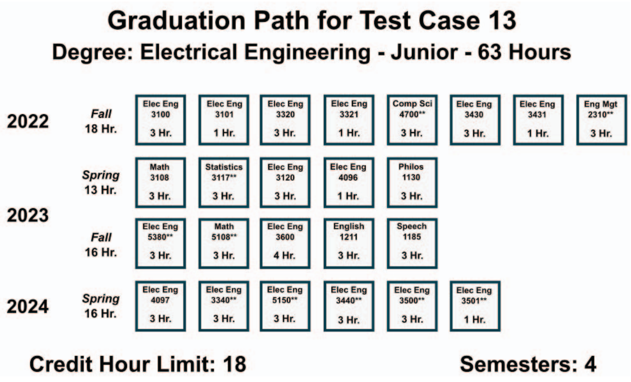


Fig. 12. Recommended path for test case 13.

fulfilled all pre/corequisite and program requirements were recommended in every test case. If the student's interest were known, the algorithm was able to incorporate a maximum number of courses of interest into the recommended path. The upper bound is determined by the constraints set by each degree program on elective courses.

Looking past individual recommendations, the test results show that the system offers potentially 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 all three programs, yet it can be placed in practically any semester. The recommended path for the third test case has the course in the seventh semester, while it is recommended for the first semester in the fifth test case. This means that despite the course being a basic required course, no other courses in the program require it. It could be worth examining the necessity of the requirement.

V. Conclusions

This paper presented a recommendation algorithm that considers a student's academic history and declared interests along with program requirements to determine an optimal semester-by-semester course schedule to graduation. The potential for reducing time-to-degree is of special note, as it can increase retention and graduation rates. The ability to set a desired number of credit hours per semester allows students to set their own pace. A more balanced load from semester to semester can also be helpful. 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. Students who graduate earlier will enter the workforce earlier, positively affecting the national economy and filling high-demand job openings, which is especially important in the current labor shortage.

There are several avenues for future contributions that build upon this work. The recommendation algorithm can easily be generalized to consider requirements for minors, graduate degrees, and accelerated programs that allow students to earn concurrent bachelor's and master'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, which could necessitate new requirement formats. A second and more significant extension to this work involves development of a machine learning system that collaborative filtering to determine acceptable credit hour limits for a given student, or to recommend course schedules with consideration of the probable performance of the student on each course.

References

[1] National Academy of Engineering, "Grand challenges for engineering," http://www.engineeringchallenges.org, Tech. Rep., 2008, accessed: 02-01-2023.

[2] M. L. Bernacki, M. J. Greene, and N. G. Lobczowski, "A systematic review of research on personalized learning: Personalized by whom, to what, how, and for what purpose(s)?" *Educational Psychology Review*, vol. 33, no. 4, pp. 1675–1715, Dec. 2021.

[3] S. Patrick, K. Kennedy, and A. Powell, "Mean what you say: Defining and integrating personalized, blended and competency education," http://www.inacol.org/, Intl Association for K-12 Online Learning (iNACOL), Tech. Rep., Oct. 2013, accessed: 02-01-2023.

[4] H. Ma, X. Wang, J. Hou, and Y. Lu, "Course recommendation based on semantic similarity analysis," in *Proc IEEE Intl Conf on Control Science and Systems Engineering (ICCSSE)*, 2017, pp. 638–641.

[5] M. Premalatha and V. Viswanathan, "Course sequence recommendation with course difficulty index using subset sum approximation algorithms," *Cybernetics and Information Technologies*, vol. 19, pp. 25–44, Sep. 2019.

[6] Q. Hu, A. Polyzou, G. Karypis, and H. Rangwala, "Enriching course-specific regression models with content features for grade prediction," in *Proc IEEE Intl Conf on Data Science and Advanced Analytics (DSAA)*, 2017, pp. 504–513.

[7] M. Srifi, B. A. Hammou, A. A. Lahcen, and S. Mouline, "A concise survey on content recommendations," in *Proc Intl Conf on Big Data, Cloud and Applications*, 2018, pp. 393–405.

[8] G. Borges and I. Stiubiener, "Recommending learning objects based on utility and learning style," in *Proc IEEE Intl Frontiers in Education Conf (FIE)*, 2014, pp. 1–9.

[9] Y. Zhu, J. Mu, Q. Tang, J. Wang, H. Li, Z. Li, X. Wang, H. Yang, and Y. Gao, "Application of intelligent course selection recommendation system based on ipv6," in *Proc IEEE Intl Conf on Software Engineering and Service Science (ICSESS)*, 2018, pp. 1–5.

[10] X. Yu, D. Wei, Q. Chu, and H. Wang, "The personalized recommendation algorithms in educational application," in *Proc IEEE Intl Conf on Information Technology in Medicine and Education (ITME)*, 2018, pp. 664–668.

[11] A. Klašnja-Milićević, B. Vesin, M. Ivanović, Z. Budimac, and L. C. Jain, "Recommender systems in e-learning environments," in *E-Learning Systems: Intelligent Techniques for Personalization*, ser. Intelligent Systems Reference Library. Springer Intl Publishing, 2017, no. 112, pp. 51–75.

[12] M. C. Urdaneta-Ponte, A. Mendez-Zorrilla, and I. Oleagordia-Ruiz, "Recommendation systems for education: Systematic review," *MDPI Electronics*, vol. 10, pp. 1–21, Jul. 2021.

[13] H. He, Z. Zhu, Q. Guo, and X. Huang, "A personalized e-learning services recommendation algorithm based on user learning ability," in *Proc IEEE Intl Conf on Advanced Learning Technologies (ICALT)*, 2019, pp. 318–320.

[14] K. Bhumichitr, S. Channarukul, N. Saejiem, R. Jiamthapthaksin, and K. Nongpong, "Recommender systems for university elective course recommendation," in *Proc Joint Conf on Computer Science and Software Engineering (JCSSE)*, 2017, pp. 1–5.

[15] K. Ganeshan and X. Li, "An intelligent student advising system using collaborative filtering," in *Proc IEEE Intl Frontiers in Education Conf (FIE)*, 2015, pp. 1–8.

[16] T. Morrow, A. R. Hurson, and S. Sedigh Sarvestani, "Algorithmic support for personalized course selection and scheduling," in *Proc IEEE Intl Computers, Software, and Applications Conf (COMPSAC)*, 2020, pp. 143–152.

[17] G. L. Heileman, C. T. Abdallah, A. Slim, and M. Hickman, "Curricular Analytics: A framework for quantifying the impact of curricular reforms and pedagogical innovations," 2018. [Online]. Available: https://arxiv.org/abs/1811.09676

[18] J. Xu, K. H. Moon, and M. van der Schaar, "A machine learning approach for tracking and predicting student performance in degree programs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 5, pp. 742–753, Apr. 2017.

[19] A. Hellas, P. Ihantola, A. Petersen, V. V. Ajanovski, M. Gutica, T. Hynninen, A. Knutas, J. Leinonen, C. Messom, and S. N. Liao, "Predicting academic performance: A systematic literature review," in *Proc ACM Conf on Innovation and Technology in Computer Science Education*, 2018, p. 175–199.

[20] Google, "Google OR tools," https://developers.google.com/optimization/cp/cp_solver, 2022, accessed: 02-01-2023.

[21] "Missouri University of Science and Technology Course Catalog," https://catalog.mst.edu/undergraduate/courselist/, 2022, accessed: 02-01-2023.