Summer 2024

Learn from the Past: using Peer Data to Improve Course Recommendations in Personalized Education

Colton Walker
Missouri University of Science and Technology

Department:

LEARN FROM THE PAST: USING PEER DATA TO IMPROVE COURSE RECOMMENDATIONS IN PERSONALIZED EDUCATION

by

COLTON MICHAEL WALKER

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 SCIENCE

2024

Approved by:

Sahra Sedigh Sarvestani, Advisor
Ali R. Hurson, Co-Advisor
Dr. Patrick Taylor

ABSTRACT

This work describes a recommendation approach designed to enhance student success by identifying semester schedules and graduation paths. The primary objective is to provide personalized graduation path recommendations rooted in individual student performance and draw insights from the academic journeys of similar students who successfully graduated. The original research contribution of this work lies in the development of a graduation path recommender system that leverages a combination of Markov Decision Process, Q-Learning, and collaborative filtering techniques to pinpoint graduation paths with a higher likelihood of leading students to success based on their academic progress thus far. The effectiveness of the proposed approach will be verified through its use of the previous student data, and course data within different departments such as Computer Science and Computer Engineering.

The proposed approach distinguishes itself through its utilization of collaborative filtering to identify similar students and understand how they achieved success in their graduation journeys and Q-learning to identify the semester load that leads a student to academic success. Moreover, it can be adapted to various disciplines provided there is a sufficient amount of data available to generate tailored pathways. This recommendation framework has been successfully implemented within the Pervasive Cyberinfrastructure for Personalized eLearning and Instructional Support (PERCEPOLIS). PERCEPOLIS has been specifically designed to support student success by offering guidance in course selection and graduation pathways, with the goal to improve retention rates, enhanced student performance, and increase graduation rates.

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude to all those who have been instrumental in supporting and guiding me throughout my educational journey. First and foremost, I want to express my profound love and appreciation for my family, whose unwavering support has been an invaluable source of strength and motivation throughout my entire educational career. I am also deeply indebted to my advisors, professors, and peers, who have generously provided me with numerous opportunities to learn, grow, and develop as a student. Their guidance, encouragement, and expertise have played a pivotal role in shaping my academic pursuits and personal development.

I would like to acknowledge the continuous encouragement and mentorship I received from Dr. Sedigh and Dr. Hurson, whose expertise and guidance were instrumental in the successful completion of this thesis.

In addition, I would like to extend my thanks to my friends and colleagues who have supported me in countless ways, whether through stimulating discussions, sharing resources, or simply offering a friendly ear when needed. This academic journey would not have been possible without the support of countless individuals, and for that, I am truly grateful. Thank you all for your unwavering encouragement and belief in my abilities, which have made this thesis a reality.

This work is supported, in part, by the National Science Foundation (NSF) under Award DUE-1742523.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

1. INTRODUCTION

Education, particularly within the realm of higher education, has progressively become intertwined with technology. This integration of technology has left its mark on various facets of university life, encompassing not only the classroom and coursework but also numerous other domains. Nevertheless, certain responsibilities such as academic advising, and more precisely, the process of students selecting their courses, have thus far failed to fully harness the advantages that technology can offer. In short, the strategies employed by students when opting for courses in a given semester have remained largely unchanged over the span of several decades.

Conventionally, a student engages in a sequence of steps, including conducting a degree audit, consulting their advisor, and finalizing their course selection for the upcoming semester. Often, little attention is given to meticulously strategizing and pinpointing courses and schedules that would serve the student's goals, be it in terms of timely graduation, financial considerations, personal enjoyment, workload distribution, or even the level of challenge. This tendency is perfectly reasonable, as delving into the unique portfolio and graduation trajectory of each student is a formidable and nearly impractical endeavor, given the multitude of potential courses and viable pathways to completion. Consequently, this situation can lead students to experience a sense of being inundated by their course load, weighed down by academic pressures, and in some instances, it might even culminate in academic setbacks for certain students.

The focus of this research lies in capitalizing on the accessibility afforded by computational processing power and available data. The aim is to extract insights from trends in academic performance and subsequently apply these insights to empower students in formulating long-term plans, as well as conducting what-if analyses. These endeavors are geared toward the meticulous selection of courses, ultimately guiding students toward achieving graduation while maintaining a commendable GPA.

Prior to this work, different recommendation frameworks existed, some focusing on constructing a full path to graduation while others provided semester-based course recommendations; each with their own limitations. Full-path graduation recommendations prioritized degree requirements but lacked personalization. On the other hand, semester-based recommendations considered student and course data but only provided guidance for the upcoming semester without a holistic view of a graduation path.

Our research bridges these gaps by proposing a recommendation framework that combines the strengths of both approaches. We aim to recommend a graduation path that satisfies degree requirements while also tailoring schedules based on past student data, enhancing student academic performance. This framework leverages a wealth of data encompassing a student's academic history, degree program requisites, available courses, and the achievements of past graduates. The central objective remains to generate semester-by-semester course schedules that chart a coherent path to graduation, taking into account specific criteria such as graduation timeline and projected GPA for each semester.

The original research contribution of this thesis is in the recommendation framework, which is centered on a Markov decision process augmented by collaborative filtering to generate several potential graduation paths and then identify the path most conducive to the student's success, as measured by their past performance, such as GPA.

This endeavor falls within the purview of the PERCEPOLIS cyberinfrastructure for personalized education [1] [2]. The author's contribution encompasses the intricacies of PERCEPOLIS, ranging from its design to its implementation and prototype development. The research presented in this paper marks a pivotal and substantial advancement within the context of the recommendation framework employed by PERCEPOLIS. Notably, the integration of reinforcement learning introduces an innovative dimension, enabling the estimation of potential student performance through the analysis of analogous student data. In a recent publication concerning PERCEPOLIS, an in-depth account is provided of the previously utilized multi-objective optimization approach for path recommendation towards graduation [2].

The distinctiveness of the proposed approach resides in its incorporation of reinforcement learning and historical data to suggest course schedules that have proven advantageous for comparable students. This methodology exhibits a high degree of intelligence and flexibility, learning from its environment and factoring in numerous facets to tailor its recommendations. The personalized course schedules that emerge from this approach hold the promise of amplifying a student's accomplishments and curtailing both the financial burden and the duration of their degree pursuit. On an institutional scale, this transformative approach has the potential to foster heightened student retention and graduation rates.

This thesis will delve into the background of the proposed approach, examining the spectrum of literature in this area to highlight where successes have been achieved and where further investigation is required. This research aims to bridge these gaps. Furthermore, the PERCEPOLIS section will

elaborate on the system that serves as the foundation for the algorithm's implementation, detailing its architecture and significance. The methodology section will provide an in-depth explanation of how the novel algorithm functions, outlining the purpose of each component and the anticipated results. Subsequently, the approach's validation will be presented through various test cases that compile the algorithm's key features in an accessible manner. Validation also includes ensuring that all generated paths will enable a student to graduate. Finally, the conclusion will summarize the key learnings and reaffirm the claims made earlier in this work.

2. BACKGROUND

Understanding the landscape of a research area is crucial not only for the researcher but also for the readers. By offering a detailed overview of related works and foundational concepts within this field, readers are better prepared to appreciate the novel concepts and methodologies introduced in this research. To this end, considerable effort has been invested in conducting a comprehensive analysis of significant features related to these works. This includes highlighting where previous research has succeeded and identifying areas that require further investigation. Such an approach sets the stage for my contribution, which aims to address these identified gaps and advance the field. Furthermore, this section will delve into more complex concepts pertinent to this study, providing a foundational understanding that can be leveraged for more sophisticated applications of these topics. Overall, this section serves as a critical stepping stone, laying the groundwork for the comprehensive exploration presented in this research.

Recommendation systems have become increasingly popular in e-learning as they play a crucial role in identifying valuable information related to student success in education. These systems come in various forms, all with the common goal of enhancing student performance and personalization in course selection. By leveraging a user's profile, preferences, history, and similarities to other users, these systems intelligently suggest items of interest. However, to ensure their effectiveness, it is vital to conduct a thorough review of recommendation systems in education, focusing on how they support educational practices, the developmental approach employed, and the nature of the items being recommended. Such evaluations are essential for refining and optimizing these systems, ultimately leading to improved educational outcomes and student experiences [3, 4].

Before delving into the related works, it's crucial to understand key concepts utilized in this research, namely Markov Decision Processes (MDPs), Q-Learning, and the similarity metric known as the Pearson Correlation Coefficient.

A Markov Decision Process (MDP) provides a framework for decision-making in complex systems, enabling an agent to perform actions, transition to new states, assess the outcomes, and proceed indefinitely or until a terminal state is reached. The fundamental components of an MDP include states, actions, rewards, transitions, and policies. For instance, consider a robot vacuum cleaner where the state might be its location in a room, labeled A or B, each potentially clean or dirty. The robot's actions could include cleaning the current location or moving. Rewards are assigned based on the action's effectiveness, such as a positive reward for cleaning a dirty area. Transitions

detail the likelihood of moving between states, while the value function assesses the desirability of each state—cleaning a dirty area yields a positive reward. The policy dictates the robot's behavior, such as cleaning when in a dirty area and moving otherwise.

Q-learning, a model-free reinforcement learning algorithm, extends the MDP framework to learn the value of state-action pairs, i.e., the rewards associated with actions in specific states. Using the robot example, imagine a 2x2 grid representing clean and dirty areas, with actions to either clean or move. Initially, the values in this "Q-table" are unknown; the robot must explore its environment to learn these values based on the rewards received, thereby approximating the Q-function for informed decision-making. This guides the robot towards the most efficient cleaning strategy based on the given rewards.

Similarity metrics play a pivotal role in identifying similarities within vast datasets, particularly in determining how closely users' choices align within a system. This study employs the Pearson Correlation Coefficient (PCC) as its similarity metric, valuable for assessing pairwise similarities between users and items. The PCC outputs a value between -1 and 1, where a negative correlation indicates divergent preferences (e.g., if user A rates a movie highly and user B rates it poorly), and a positive correlation signifies similar tastes. By identifying users with similar PCC scores, it becomes possible to recommend content one user likes to another with similar preferences, enhancing personalized recommendations.

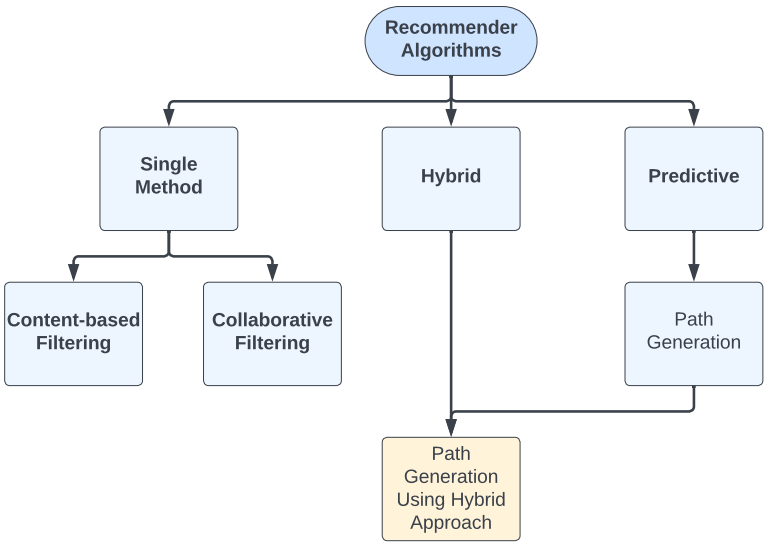

Figure 2.1. Related Work Taxonomy Figure

This research builds upon the PERCEPOLIS cyberinfrastructure [2], which focuses on personalized education by identifying suitable course schedules using multi-objective optimization. In contrast to the earlier efforts, which considered academic history, interests, and degree requirements, this paper extends the research by introducing the prediction of a student's academic performance through collaborative filtering and reinforcement learning, enhancing the course schedule recommendations for graduation.

To this extent, the review of work relating to recommendation, predictive, and path-generation approaches offers valuable insights into the broader landscape of research in recommendation systems for education, as demonstrated in Figure 2.1. By examining existing methodologies, we gain a comprehensive perspective on where the research fits in the larger picture of enhancing educational recommendation systems.

2.1. SINGLE METHOD RECOMMENDATION APPROACHES

Several works adopt one of two predominant approaches: *content-based filtering* and *collaborative filtering*.

2.1.1. Content-based Filtering. Content-based filtering (CBF) generates recommendations by analyzing item features and individual user preferences. It utilizes keywords to describe each item and creates a user profile from direct input from a user or past interactions. This method assumes that user preferences remain relatively stable over time. For course recommendation, CBF considers a student's previous course searches or input from surveys [5]. For instance, Morsomme et al. [6] employed a content-based approach to suggest courses aligned with a student's academic interests while providing cautionary alerts for potentially challenging courses based on the student's academic background.

2.1.2. Collaborative Filtering. In collaborative filtering (CF), recommendations are generated based on similarities between users and/or items. User profiling is derived from direct input or historical interactions, enabling the identification of users with analogous profiles. Enabling the discovery of interesting courses for a user by analyzing their interaction with a system. Hidayat et al. [7] utilized CF to filter information, collecting ratings, assessments of learnings, frequency of selection, and time spent on learning content. This data formed a Learner Learning Object Rating (LLOR) matrix, enabling personalized learning content recommendations. Klašnja-Milićević et al. [8] proposed a distinctive approach to enhance CF in e-learning environments by incorporating

tag-based recommendations. Their focus lies in selecting appropriate collaborative tagging techniques to drive improved, personalized recommendations aligned with learners' interests, learning styles, demographic characteristics, and prior knowledge.

An issue that arises from CF is the data sparsity problem. Data sparsity within e-learning systems refers to the scarcity of available user-item interaction data, which hinders the accurate generation of personalized recommendations. To address this issue, Pang et al. [9] introduced a Multi-Layer Bucketing Recommendation (MLBR). MLBR transforms learner vectors into a uniform length dimension and organizes them into buckets containing similar users, enabling more effective course recommendations. This approach not only improves issues related to data sparsity but also reduces the time cost for generating recommendations. Another effort to mitigate data sparsity was made by Liu, Xiuju [10], who developed Collaborative Filtering based on Influence Sets of the E-learning Group's Behavior (CF-ISEGB). CF-ISEGB evaluates the target object by simultaneously combining the k nearest neighbors and k' reverse nearest neighbors, effectively overcoming limitations in existing CF-based algorithms. This results in more accurate predictions for the target object and enhances the evaluation density of target objects, especially in sparse datasets.

While CF has various advantages in generating recommendations, newer and more advanced alternatives, such as the hybrid approach, have emerged. The hybrid approach combines multiple recommendation methods, mitigating the negative effects of individual approaches. In contrast to these approaches that solely relied on CF, the proposed method incorporates several approaches, resulting in enhanced recommendations surpassing the capabilities of single-method approaches.

2.2. HYBRID APPROACHES

Hybrid approaches refer to the integration and combination of multiple recommendation techniques and algorithms to provide more accurate and personalized recommendations for learners. The aim is to combine the strengths of different approaches while mitigating their weaknesses, leading to an improved learning experience for the users. In the scope of a hybrid approach, several approaches have been proposed, including Collaborative Filtering, Content-based Filtering, Knowledge-based Systems, Matrix Factorization Techniques, Context-aware Recommendations, and Reinforcement Learning approaches, resulting in numerous combinations.

Jing Li et al. [11] developed a content-based collaborative filtering recommendation algorithm that utilizes a scoring matrix to compute similarities between users or items, facilitating personalized recommendations based on these similarity relationships. The authors found that the integrated

recommendation system efficiently assists users in discovering high-quality information that aligns with their interests, ultimately saving valuable time and costs. Jerson Rivera [12] developed a hybrid recommender system for enrolling in elective subjects for engineering students. The system utilizes both content-based and collaborative filtering approaches to analyze students' academic behavior and relationships. Additionally, machine learning models were employed to generate recommendations based on the similarity between subject contents and elective subject lines, as well as the academic relationships among students.

Furthermore, Gongwen Xu et al. [13] developed a system that combines knowledge graphs and collaborative filtering. By integrating the semantic similarity of recommended items into the calculation, this approach achieves improved precision and recall, leading to more accurate and efficient course recommendations. However, it faces challenges such as user interest drift and the time effectiveness of historical data. In another study, Zameer Gulzar et al. [14] devised a hybrid approach focusing on N-gram query classification, expansion-based information retrieval, and ontology support for course recommendations. Their proposed system proves to be more suitable, effective, and beneficial to learners compared to single-method approaches.

Considering the aforementioned hybrid approaches, certain limitations persist in terms of the user's options with the provided recommendations. To enhance this aspect, predictive approaches step in, offering valuable insights through their predictions, including student performance, likability, and other relevant factors. Such predictive approaches not only provide greater user understanding but also come in various forms, presenting an array of possibilities for improving the overall recommendation experience.

2.3. PREDICTIVE APPROACHES

Predictive approaches represent the next natural progression of recommendation systems, offering valuable insights to both users and the recommending system. By generating informed predictions, these systems can make informed decisions on tailored courses, learning objects, or other e-learning recommendations. Hellas et al. [15] identified predictive approaches into four main categories: Classification, Clustering, Mining, and Statistical. This classification framework contributes to a comprehensive understanding of diverse predictive methodologies and their applications.

Classification methods, relying on supervised learning, play a pivotal role in predictive approaches. Noteworthy examples of classification methodologies are demonstrated in the works of Huda Al-Shehri et al. [16], who utilized a Support Vector Machine and K-nearest neighbor for

prediction, and N. Buniyamin et al. [17], who employed educational data mining to predict and classify engineering students. These studies showcase the effectiveness of classification techniques in enhancing predictive capabilities within various domains.

Clustering methods, based on unsupervised learning, constitute another vital aspect of predictive approaches. Notably, V. K. Anand et al. [18] developed a machine learning approach utilizing recursive clustering to group students in a programming course according to their performance. Subsequently, a linear regression technique was employed to assess the student's performance, leading to customized learning content for individual students. Building upon this, Wala Bagunaid et al. [19] introduced the AISAR: Artificial Intelligence-Based Student Assessment and Recommendation System. This system encompassed score estimation, clustering, performance prediction, and recommendation. By utilizing a recurrent neural network, a spatial clustering algorithm, and a threshold-based MapReduce, the system effectively predicted student performance, thereby generating personalized recommendations. These studies exemplify how predicting student performance can effectively drive tailored recommendations for users.

Data mining methods, focused on identifying frequent patterns or feature extraction, present another facet of predictive approaches. For instance, Zahyah Alharbi et al. [20] developed a data mining technique to predict poor student outcomes. However, their primary objective was not to recommend course objects based on these predictions. Instead, they utilized the predictions to identify struggling students and offer remedial sessions to improve their performance. Similarly, within educational data mining, Mustafa Yağcı [21] devised a machine learning algorithm to predict students' academic performance, using midterm exam grades as input data. This approach, employing various machine learning algorithms, achieved an accuracy of 70-75 percent in predicting student final exam grades. While these methods are valuable for predicting student performance within the scope of a course, they do not directly aid in predicting overall course grades, which is essential for course recommendations.

Finally, statistical models, utilizing correlation, regression, t-testing, and other techniques, stand out as the most commonly observed methods for predicting student performance [15]. For instance, Hu et al. [22] proposed a course-specific regression model that considers information about students, courses, and instructors to predict student performance. Although their system successfully predicted student performance using data from public universities, its focus was on enhancing the performance of statistical models through content features rather than incorporating course recommendations. Another statistical model was developed by Pei-Chann Chang et al. [23], utilizing

a hybrid approach of collaborative filtering and artificial immune systems (AIS). By employing cosine similarity, Karl Pearson correlation, and AIS clustering, they predicted the scores of 3rd and 4th-year students. While this method offered a hybrid approach and made recommendations based on score predictions, it was limited to 3rd and 4th-year students, failing to provide personalized recommendations for younger students.

2.4. PATH GENERATION APPROACHES

Path generation is a facet of predictive approaches, encompassing the formulation of a sequential route toward a specific goal. In the academic context, this process involves devising a predictive path to graduation, considering both the essential courses required for completion and the necessary prerequisites for each course. By carefully mapping out these interconnected steps, individuals can efficiently navigate their educational journey and achieve their desired objectives.

Various other approaches have been explored within the realm of course sequence recommendations. For instance, M. Premalatha and V. Siswanathan [24] proposed a Course Sequence Recommendation using Subset Sum Approximation Algorithms. This method generated course recommendations based on mandatory courses for a degree. However, it lacked consideration for elective recommendations for students and did not involve predicting student performance. Instead, it focused on estimating course difficulty based on a student's past performance.

Another approach, pioneered by Jie Xu et al. [25], involved finding course sequence recommendations utilizing a Forward-Search Backward-Induction algorithm. This resulted in a shorter time to graduation and employed an online regret minimization learning algorithm to maximize a student's GPA. While this method aimed to enhance student performance by predicting their academic outcomes, it solely relied on a student's past performance in prerequisites, neglecting insights from other students' performances when making recommendations.

In the current state, as discussed in Section 1, the PERCEPOLIS project successfully offers course pathway recommendations leading to graduation [2]. The recommender system adopts a pervasive personalization approach, optimizing a student's journey through a multi-objective optimization problem. The primary objectives are to minimize the time-to-graduation, credit hours taken, and to align the pathway with the student's chosen career trajectory. While this solution effectively recommends course pathways to students, it cannot predict a student's performance based on the given recommendations.

This section introduces the Markov Decision Processes (MDP), Q-learning, and Pearson Correlation Coefficient (PCC). Understanding these components in a broad sense is crucial, as their applications will be explored in greater depth throughout the methodology. Furthermore, this segment delves into a comprehensive review of related literature and its contributions to the domain, particularly focusing on course recommendation systems. It highlights the superiority of hybrid models over single-method strategies, which underpins the rationale for adopting a hybrid approach in this research. The analysis also underscores the value of predictive models in providing learners with insights, enabling them to make well-informed course selections. Nonetheless, it identifies a gap in the creation of complete educational pathways that offer ongoing predictions to further aid students, which this research aims to address. By integrating predictive modeling and an MDP framework, my methodology introduces a novel hybrid approach to generate complete academic pathways, from course selection to graduation, thereby facilitating informed decision-making for students.

3. PERCEPOLIS

The PERCEPOLIS cyberinfrastructure is a powerful tool designed to enhance graduation rates, student retention, and academic performance. This web application offers students a comprehensive pathway to graduation, considering various essential factors such as time to degree completion, likelihood of success, individual interests, and career aspirations. By enabling students to generate a personalized academic path before advising meetings, it empowers them to find a semester schedule that aligns with their current university progress, in terms of completion to degree, while also taking their long-term graduation goals into account. This stands in contrast to the existing method, which solely involves running a degree audit and disregards the critical personalization factors crucial for student success.

PERCEPOLIS strives to make smart decisions based on a student's current degree progress and the various degree requirements. To achieve this, a web architecture has been established to manage the system's requirements and run the recommendation algorithms. This architecture is divided into two components: the front end and the back end, each handling distinct tasks for PERCEPOLIS. The overall architecture can be seen in Figure 3.1.

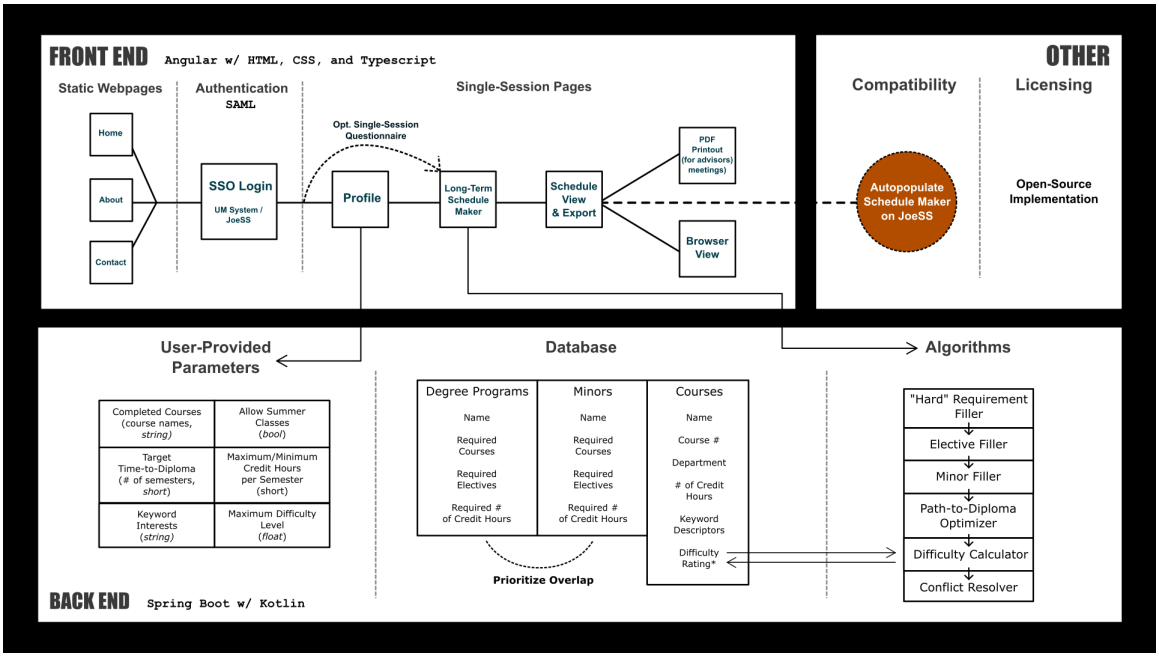

Figure 3.1. Functional Diagram of PERCEPOLIS Architecture

3.1. WORK TO DATE

This project has been developed for many years and has encompassed several different approaches to obtain its goal. Earlier work reported in [26] developed a user interface (UI) to obtain student information, send it to the back end, and generate a semester schedule for that student. Additional effort has been published [2] by Nicolas Dobbins et. al. and focused on the development of a recommendation system that used university data and degree requirements to generate paths to graduation. This work used a multi-objective optimization approach to identify courses based on student interest while reducing time to a degree given credit hour limits.

3.1.1. Front-end. The current status of the front end encompasses numerous features that facilitate the system's comprehensive functionality. This enables an interactive platform accessible to students, advisors, department heads, and the registrar, each with varying levels of access privileges. It promotes an engaging environment for communication, viewing schedules, and tailoring features to meet individual needs, shown in Figure 3.2. For instance, students can readily access their recommended schedules, while an illustrative graduation path for a computer science major is depicted in Figure 3.3.

The front-end implementation has been completed and seamlessly integrated into the system. It was developed using a combination of essential web technologies, including HyperText Markup Language (HTML) and Cascading Style Sheets (CSS), both fundamental for any web-based application. In addition, Angular was chosen for its features, such as the creation of custom and reusable components, enhanced productivity, consistent code structure, and straightforward testing capabilities. Angular's compatibility with TypeScript, an open-source language closely related to JavaScript but distinguished by its static typing, further contributed to the development process. This combination of languages facilitated the construction of web pages, many of which borrowed elements from one another. Furthermore, it effectively manages server requests, enabling users to send requests and view the corresponding responses on the page.

3.1.2. Communication between Ends. The communication component of the full-stack application was established through the use of SPRING Boot and a REpresentational State Transfer (REST) API. SPRING Boot, a Java framework, simplifies the execution of Java-based applications, making it a valuable part of this project. Likewise, the implementation of a REST API adheres to a set of standards governing inter-application communication, particularly between the front-end and back-end systems. The RESTful API permits essential operations like GET, POST, PUT, DELETE,

and more, which are fundamental in any system architecture. These components were chosen for their compatibility with both front-end and back-end software, making them vital for the seamless functioning of the application as portrayed in Figure 3.4.

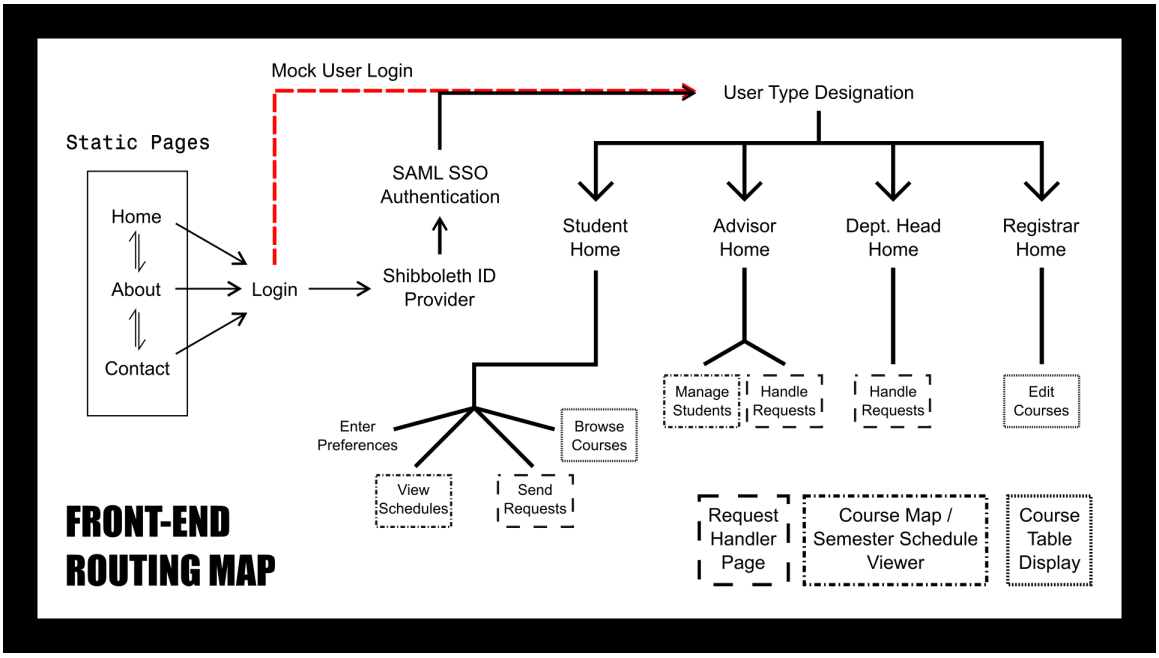

Figure 3.2. Front-end Routing for each User Type

3.1.3. Back-end. The current state of the back end involves multiple components working cohesively to ensure the smooth execution and compilation of the supported algorithms. To maintain this functionality, the back-end was developed using Kotlin, a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin, while Java-based, offers a more concise syntax than Java, reducing the amount of code required for the same operations and enhancing readability, which is particularly beneficial for new developers.

The back-end also relies on a database system to store the data essential for the algorithms. PostgreSQL, a database management system (DBMS) renowned for its robust data consistency and integrity, is employed for this purpose. Specifically through the use of MicroStream, an object

Figure 3.3. Sample Graduation Path Generation

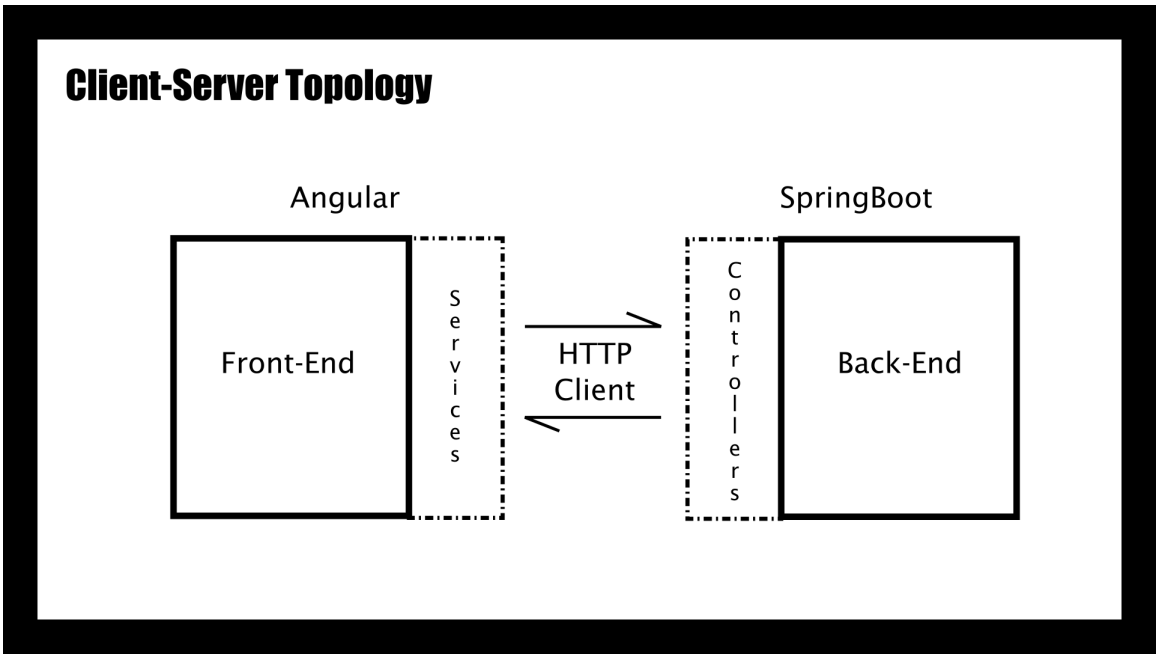

Figure 3.4. Controller-architecture

graph persistence engine built for the Java Virtual Machine (JVM). Within the system, two primary databases are managed. The first contains information about courses and their underlying structure, including course IDs, descriptions, departments, credit hours, course equivalencies, and other relevant details, shaping the foundation of courses within the educational organization. The second database holds anonymized student data, encompassing courses taken, degree pursuits, associated grades, degree requirements, and any other additional information pertinent to their graduation progress.

3.1.4. Algorithms. Several algorithms have been developed and are currently within the PERCEPOLIS environment. This includes:

i) a multi-objective algorithm that creates a path to graduation as depicted in Figure 3.5. This algorithm can work with a student's given transcript, degree, interest, and semester credit hour limit. It generates a semester-by-semester path to graduation which is viewable on the front-end for the student [2].

ii) a Topic Modeling algorithm that uses Latent Dirichlet Allocation (LDA) through Gibbs Sampling that given student interest through keywords would generate a list of courses that are found to be most accosted with those keywords. As such, it would allow a student to view those courses while adjusting their schedules to understand which ones are within their interest.

The forthcoming addition to PERCEPOLIS introduces a novel algorithm developed within this thesis. At present, PERCEPOLIS lacks a method that employs collaborative filtering and machine learning to schedule courses for a semester and conjoin those into a full graduation path. This approach utilizes historical data to identify paths that previous students with similar academic progress have pursued. The underlying concept is that students who share similar course histories and academic performance are likely to continue performing similarly in the future. Consequently, recommending a semester that proved successful for a similar student, particularly in terms of GPA, is expected to contribute to improved academic performance and a more efficient journey to graduation.

Furthermore, the machine learning aspect is harnessed to assess course loads that have led to the highest success rates relative to the number of credits taken. The overarching objective is to integrate these approaches and identify a graduation path that maximizes a student's likelihood of success. In essence, this approach offers a counterbalance to the previous optimization strategy, which primarily focuses on expediting time to graduation. While this method doesn't guarantee the quickest graduation path, it utilizes historical data to recommend successful paths previously traversed by students, providing a valuable alternative for academic success.

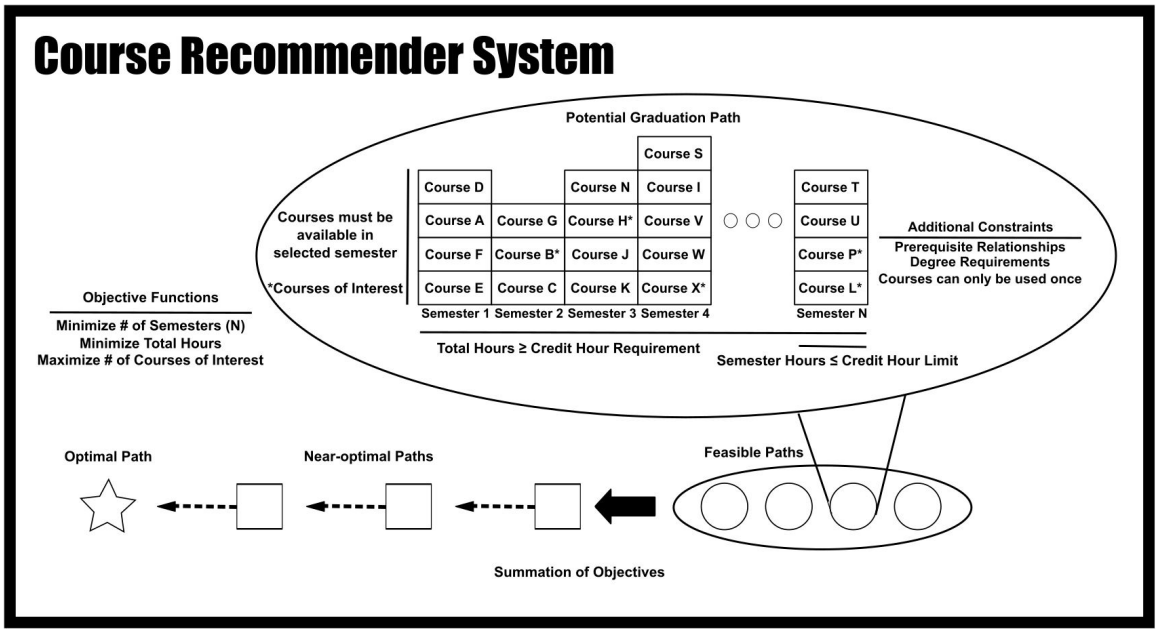

Figure 3.5. Optimization Algorithm Architecture

3.2. SUMMARY

The current state of PERCEPOLIS is pivotal, as it currently empowers students to chart their path to graduation. This is a noteworthy improvement, as previously, students had to manually complete a degree audit, select courses for the semester, and then consult with their advisor to finalize their choices. Automation of this process, driven by students' preferences and information, seamlessly integrates academic advising and provides a powerful tool for students to make informed semester and graduation path selections. This optimized approach not only benefits students but also contributes to higher retention rates, improved academic performance, and ultimately, increased graduation rates.

This section serves as an overview of the PERCEPOLIS project, shedding light on its current application progress and illustrating where this research fits within the system. The project's significance lies in its current contributions and its potential for future development. Numerous opportunities exist for expanding functionality, generating excitement for the project's future prospects. With this strong foundation, PERCEPOLIS is poised to evolve into a robust platform, driving students toward greater success in their academic endeavors.

4. METHODOLOGY

In the ever-evolving landscape of education, providing personalized and effective course recommendations to students is crucial for their academic success and overall satisfaction. To this extent, Markov Decision Processes (MDPs) offer a powerful framework for modeling the decision-making process in dynamic systems. Here, we explore the application of MDPs to the domain of course recommendation, where a student's academic performance varies from one semester to the next. By leveraging historical data and the performance of past students with similar trajectories, the MDP-based recommender system generates semester-by-semester course schedules that aim to increase the probability of success for each student.

Figure 4.1 illustrates the proposed architecture of the course recommendation system. The input to this approach consists of a student's current progress toward their degree, including the courses taken and the associated grades. Additionally, a range for the credit hour limit is required to enable the algorithm to select semesters that fall within this range, if possible. The first stage of the algorithm involves initialization and a validation check. This stage checks whether a valid graduation path has been generated after each iteration and outputs the schedule if so. If not, it proceeds to the next part of the algorithm, the student filtering stage. Here, several filtering processes take place, essentially sifting through potential students to identify the most similar collection of students from which to select states. Subsequently, the semester state filtering occurs, where the next states are selected based on validity and the selection of states with a specific number of credit hours, as determined by the Q-learning function. Finally, the semester with the highest GPA is selected, appended to the student's transcript, and the validation check is performed again.

The objective is to identify paths to graduation that increase student success both through finding semester schedules in which similar peers performed academically well, and by using Q-learning to determine a semester credit load that offer the most success through GPA. Inputs to the recommendation system include the student's academic history (courses taken and grades earned), the course offerings at their institution, and anonymized data from students who have completed the same degree program.

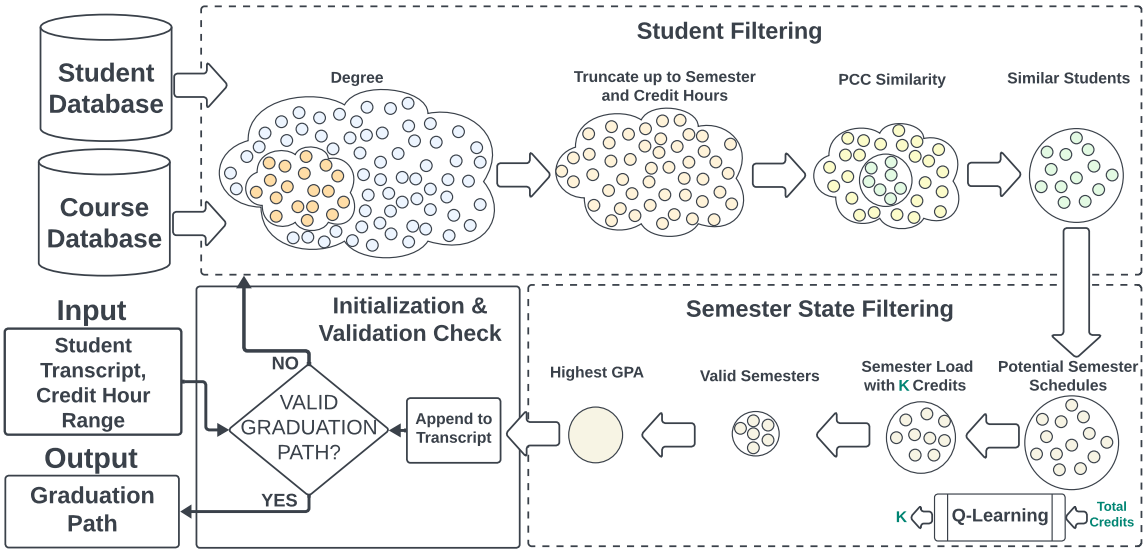

Figure 4.1. Architecture of the Course Recommendation System

The creation of the recommendation system involves the following research tasks:

1. Formulating the MDP, which includes defining the states, actions, and rewards and identifying the transition probabilities (4.1).
2. Formulating the value and policy functions, using the Q-Learning approach (4.1.4).
3. Training the reinforcement learning algorithm.
4. Validation of the results using the PERCEPOLIS platform.

4.1. MARKOV DECISION PROCESS

MDPs provide a framework for modeling the decision-making process in a dynamic system. The dynamic nature of course recommendation is because a student's academic performance changes from one semester to the next.

Figure 4.2 illustrates the Markov Decision Process (MDP) used to generate course schedules semester by semester. An MDP consists of several components, including states, actions, rewards, transitions, and policies. Upon examining the figure, we can identify many of these elements. For example, S_t denotes the current state. Actions are represented by the list of courses and credit hours, indicating the possible selections from the current state. Each action is associated with a potential reward linked to that state, determined by the possible GPA outcome. The Q-table plays a crucial

role in this MDP by reducing the number of potential states to those within a specific credit limit. To determine the credit limit for a semester, the Q-function considers the current total credit hours, consults the Q-table, and selects a credit hour limit that maximizes the reward. While the policy is not depicted in the figure, it is integral to the algorithm, guiding the selection of the next semester's schedule based on the highest future cumulative GPA. Likewise, the transition probabilities are not shown as each state is equally likely to be selected. Finally, the subsequent state reflects the updated transcript with the newly appended schedule.

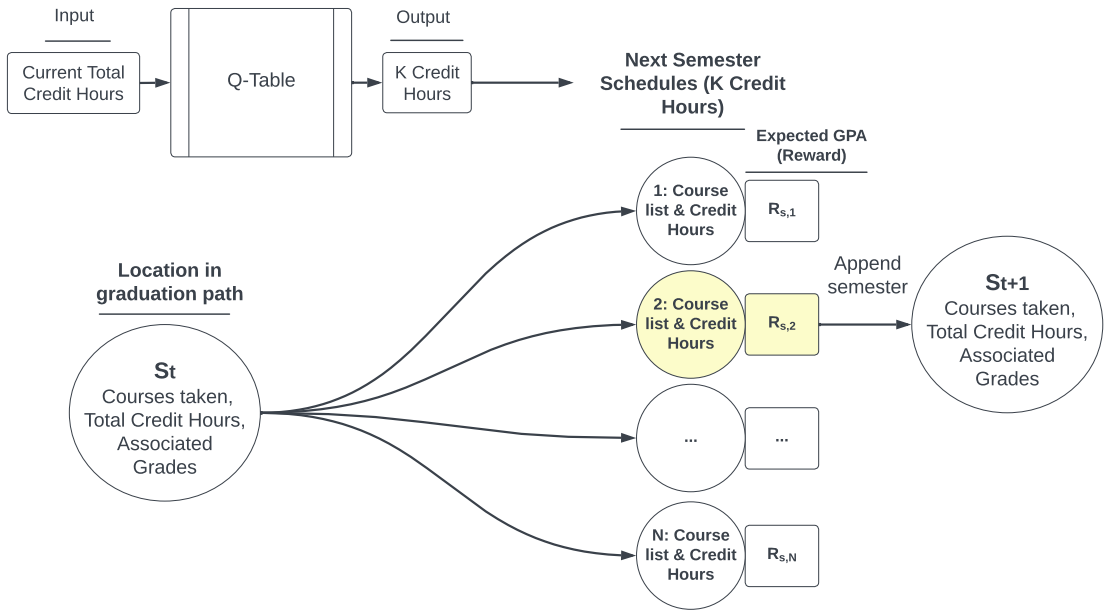

Figure 4.2. Markov Decision Process Used for Course Recommendation

4.1.1. State. In a Markov Decision Process (MDP), a state encapsulates the current situation of the system, containing all information relevant to decision-making and the determination of future outcomes. This includes various variables and factors that influence the decision process. In this specific context, the state of the MDP is defined by the current courses a student is taking, along with their corresponding grades.

When the system transitions to a new state, it incorporates newly added courses and their predicted grades. These predicted grades are estimated based on the performance of similar students, providing a data-driven approach to forecasting academic outcomes. The distinction between the current state and the subsequent state is marked by the addition of these new courses to the student's academic record. This update not only includes the anticipated grades for these courses but also the additional credit hours accumulated.

The MDP's starting state corresponds to the student's most recently completed semester on their path to graduation. This state includes details about their degree program, courses taken, and associated grades. Using this starting state, the MDP generates additional semesters, based on peer student data, and determines a unique path to graduation.

4.1.2. Actions. In a Markov Decision Process (MDP), 'actions' refer to the choices or decisions that an agent can make in a given state. These actions are potential moves the agent can take to influence the system or environment with which it interacts. Specifically, in this MDP, an action involves selecting a semester schedule for a given state. A 'semester schedule' is defined as a collection of courses that are enrollable in a given semester. Potential next states are identified based on information from students with similar academic paths who have already graduated.

Similarity is determined by several factors, including the student's degree program, the number of semesters completed, total credit hours, and courses taken. These factors ensure that the trajectory toward graduation is comparable among students.

The Pearson Correlation Coefficient (PCC) is employed to ascertain the linear correlation between an advisee's course grades and the corresponding grades of other students, considering the same semester and credit hour range. The PCC, ranging from -1 to 1, calculates the correlation between two data sets (X and Y). A value of 1 indicates the highest positive correlation, while -1 indicates a negative correlation. The correlation is computed using the values of X and Y, along with their respective means. Specifically, the function assesses the similarity between students based on the grades earned in identical courses, converting grades to numerical values (S=6, A=5, B=4, C=3,

D=2, F=1, U=0). A minimum similarity threshold of 0.4 is set, based on the PCC scale, where 0.4 to 1 indicates a moderate to high correlation. This threshold ensures a moderate selection of the next states while maintaining accuracy in predictions.

$$\mathrm{P}CC = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2(y_i - \bar{y})^2}} \tag{4.1}$$

In cases where no similar students are found after applying the PCC, a fallback strategy is employed. This strategy involves generating states by selecting courses from the same semester credit level group, creating a range of states. The predicted grades in these states are derived from a combination of the student's GPA and the average grade of all students within the group.

Next semester's schedules are then chosen from this pool of similar students, forming the subsequent states. These states inform the actions taken, based on the chosen semester credit hour limit from the Q-Table and the associated rewards, which will be elaborated upon later.

4.1.3. Transition Probabilities and Rewards. In a Markov Decision Process (MDP), a transition probability quantifies the likelihood of moving from one state to another when a specific action is taken in the current state. However, as this MDP utilizes Q-Learning, a model-free approach, transition probabilities are not explicitly calculated or predetermined. Instead, this absence of a predefined model is addressed by observing interactions with the environment and recording these observations in the Q-Table.

Furthermore, the reward of a state represents the benefit an agent receives within a system for performing a particular action in that state. In this context, the expected reward for a state is determined as the average Grade Point Average (GPA) of students who have previously followed a similar schedule.

$$\mathrm{R}_s = \overline{GPA_{sch_n}} \tag{4.2}$$

4.1.4. Value and Policy Functions. The value and policy functions are crucial in the decision-making process for selecting a state. Specifically, the value function estimates the total amount of reward an agent can expect from a given starting state, whereas the policy is the strategy employed for choosing actions.

I propose using Q-Learning to identify the best policy. Q-learning is a reinforcement learning method that employs a lookup table, or Q-Table, to determine the optimal action for a given state, utilizing the Bellman equation with state and action pairs as inputs. As the agent navigates

the environment, it updates this table, thereby tracking the expected rewards over time. In this framework, the Q-Table records the expected reward for each state-action pair, with the state defined by the number of credit hours, and the action being the recommended number of credits to take. The reward is updated according to the predicted GPA for that specific state-action pair. The Q-Table is calculated using the following equation, where $Q(s, a)$ represents the Q-value for the current state, α is the learning rate, and γ is the discount factor. The learning rate α dictates the rate at which the agent incorporates new information, and the discount factor γ determines the weight given to future rewards.

$$\mathrm{Q}_{new}(s, a) = (1 - \alpha)Q_{\mathrm{old}}(s, a) + \alpha(R_{t+1} + \gamma_{\max}Q(s', a')) \tag{4.3}$$

Consequently, the policy selects a schedule based on the largest expected reward from $Q_{max}(s)$ for a given state. $Q_{max}(s)$ identifies which semester schedule credit limit offers the highest reward, leading the agent to select the schedule that promises the highest GPA, given the determined credit hours. Upon selection, the state-action pair is updated. The original Q-value, denoted as $Q_{old}(s)$, is replaced in this process. R_{t+1} represents the expected reward for the chosen state (GPA). Additionally, the maximum Q-value of the subsequent state, $Q(s', a')$ is considered. This aspect of the algorithm allows the table to look ahead and assess the future impact of its actions.

4.2. TRAINING THE DECISION-MAKING POLICY

Since this MDP employs the Q-Learning reinforcement learning approach, the Q-Table must be updated to reflect the best decisions based on training. For accurate policy training, it was determined that data specific to each degree program should be used. Consequently, the reinforcement learning algorithm was executed on hundreds of students at random times during their degree programs, resulting in thousands of starting states for the algorithm to learn from. With this data, the algorithm updated the Q-Table and generated paths, identifying semester credit loads that offered the best reward.

In reinforcement learning, several crucial hyperparameters play important roles in training the agent's policy. These hyperparameters are epsilon (ϵ), the learning rate (α), and the discount rate (γ).

Epsilon (ϵ) is used for the exploration-exploitation trade-off. It controls the balance between exploring different actions to discover potentially better ones and exploiting actions that yield known rewards. Initially, the reinforcement learning algorithm had an epsilon value of 1, making it take

random actions. As the algorithm learned from the environment, the epsilon value decreased. This process, often referred to as epsilon decay, gradually reduces the amount of random exploration and increases exploitation of the learned knowledge. When making decisions, the algorithm selects the best state-action pair with a probability of 1 - ϵ. Eventually, epsilon reached the lower value of 0.01, meaning that the algorithm takes the best-known path 99.99% of the time when generating results for the advisee.

The learning rate (α) is a parameter used for updating the agent's policy during training. It determines how much the policy should be updated based on observed rewards and experiences from the environment. A learning rate of 0.2 was used. This was to to strike a good balance between allowing new information to significantly update the Q-values while still retaining some influence from older values, promoting a stable but flexible learning process.

The discount rate (γ) is a parameter used to discount future rewards in the agent's learning process. It represents the agent's preference for immediate rewards over delayed rewards. In path generation, the goal is twofold: to graduate and to increase the likelihood of success in the student's course load. Therefore, the discount rate was set to 0.9 to value future success in coursework as much as current success.

In the context of reinforcement learning, the discount rate (γ) plays a pivotal role in balancing immediate versus delayed rewards. Specifically, γ is used to exponentially discount future rewards, thereby determining the agent's preference for immediate gratification over future benefits. This mechanism ensures that rewards expected in the future are weighed less heavily compared to immediate rewards, with the degree of discounting inversely related to the value of γ. For instance, with a γ set at 0.9, each future time step's reward must be approximately 11.1% more valuable than an immediate reward to be considered equally desirable, as future rewards are discounted by a factor of $1/\gamma$ for each time step. This nuanced approach is particularly relevant in path generation tasks, where the objective is dual: achieving graduation and enhancing the probability of success in the student's academic endeavors. By selecting a γ of 0.9, our model strategically prioritizes near-term academic achievements while still placing substantial value on long-term success, thereby fostering a balanced approach to decision-making in the student's academic trajectory.

After training the policy, it was applied to the student being advised to provide a graduation path based on the learned behavior of past students and their successes.

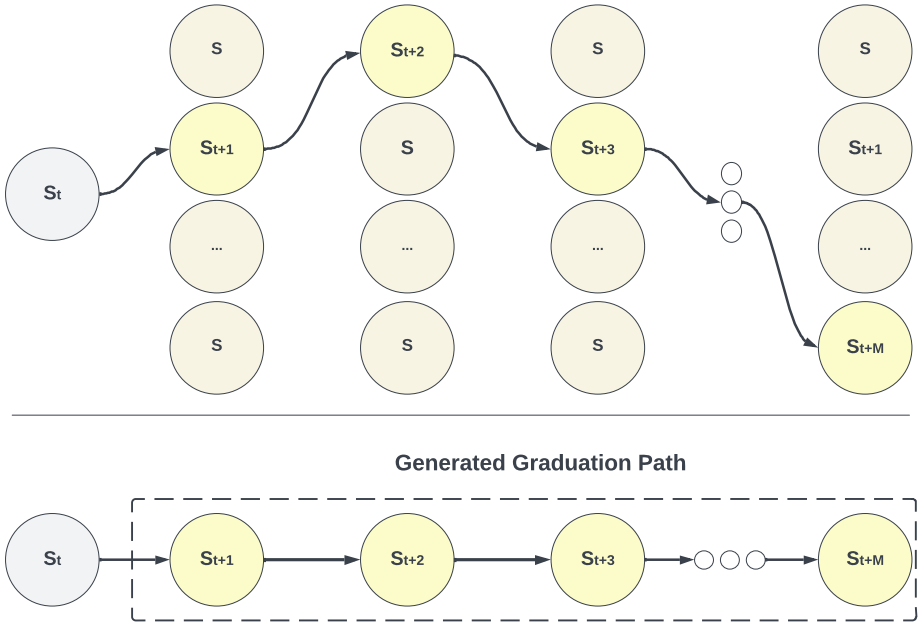

Figure 4.3. Full Path Generation

4.2.1. Path Generation. Once the policy for the reinforcement learning algorithm has been trained within a specific degree program, the path generation can proceed for the advisee. Having trained the policy, it is now capable of accurately generating a graduation path for the student, with a focus on achieving course success through GPA. To initiate the path generation, the student needs to submit their current transcript. Subsequently, the algorithm runs the MDP, selecting new states and adding them to the current transcript until a valid path has been generated as depicted in Figure 4.3. This path is then validated using the schedule generator within PERCEPOLIS, ensuring that it meets the necessary degree requirements. Finally, the student will receive a graduation path recommendation, along with an estimated GPA for the suggested path.

4.2.2. Summary. In this section, we have detailed the architecture of the system, covering both its design and implementation. The decision-making framework is centered around the use of a Markov Decision Process (MDP), which is employed not only for its applications in machine learning

but also for making informed decisions based on its training. This framework has been adapted to address the graduation path problem, specifying states, actions, rewards, transitions, and policies within this context to generate a path to graduation.

The Pearson Correlation Coefficient serves a dual purpose in this system. Firstly, it helps narrow down the potential actions for the agent by reducing the number of selectable actions. Secondly, it identifies future semesters that have been chosen by similar students—defined by the number of semesters attended, credits taken, and grades earned—suggesting that students with similar profiles are likely to perform similarly. This inference allows the system to predict the range of an advisee's performance, enabling it to make informed decisions by selecting semesters that could enhance student performance based on the highest GPA potential.

Furthermore, the application of Q-learning in determining credit hour limits has been explicitly defined. The primary aim of this methodology is to leverage historical data from peers to guide the recommendation of graduation paths, including the strategic determination of credit hour limits for each semester. This is facilitated by the Q-learning algorithm, which utilizes the Bellman equation to anticipate the outcome of future rewards for specific state-action pairs, thus inherently considering the future predicted GPA at the conclusion of their graduation path.

Overall, understanding how the system operates and how users interact with it clarifies the significance of its outputs and how to interpret these results.

5. VALIDATION AND ANALYSIS

5.1. OVERVIEW

The implementation of the path generation recommendation approach was followed by a detailed testing and validation phase. This phase involved using a variety of metrics to evaluate the algorithm's performance. The testing was conducted using a course database and a student database, spanning multiple degree programs to ensure the approach's versatility across different academic disciplines.

To assess the effectiveness of the recommendation approach, it was applied to a range of degrees at various stages of academic progression. Notably, the dataset for this analysis was sourced from students who have graduated, which reinforces the reliability of the graduation paths based on complete degree requirements.

The following sections will delve into a detailed analysis of the results obtained from this testing process.

5.2. TEST ENVIRONMENT

For validation purposes, the PERCEPOLIS environment was used to generate paths based on test cases encompassing various degree requirements and students' progress. This step involved compiling data on degree requirements, sub-requirements, course information, and student records. The recommender system was then applied to each case to verify its capability in generating practical and achievable graduation paths. Moreover, for each test case, the generation of the path took anywhere from a minute to a minute and a half. This is important as it shows that a user would be able to import their transcript and get a valid path in a reasonable time. To ensure the validity of these paths, they were cross-checked against specific degree and sub-requirements. The evaluation covered several degree programs, including Computer Science and Computer Engineering.

5.2.1. Degree Requirements. In the university system, each degree program is structured around a set of requirements and sub-requirements. Requirements are the criteria used to determine if a student has completed their degree. Within these, there are sub-requirements, which are met by completing specific courses. It is important to note that a single course cannot be used to satisfy more than one sub-requirement. To fulfill a sub-requirement, students must take designated courses that meet those particular criteria. Once all sub-requirements of a requirement are met, that requirement

is considered fulfilled. Accordingly, the completion of a degree is contingent upon fulfilling all its associated requirements. Given the vast array of degree programs, a subset was selected to evaluate the proposed method's ability to meet all requirements and generate a valid graduation path.

5.2.2. Course Data. The course data comprises a comprehensive list of all courses offered for enrollment at the university. There are a total of between 1100 and 1500 unique courses within the system each semester. Each course entry contains details pertinent to fulfilling degree sub-requirements. This information typically encompasses the semester availability of the course, its credit hour value, and the specific sub-requirement that it satisfies. These details are instrumental in determining how a selected combination of courses in any given semester can meet various sub-requirements.

Furthermore, to facilitate the generation of viable semester schedules, it is necessary for courses to meet all prerequisite requirements. A prerequisite is a course, or a set of courses, that must be completed before enrolling in a subsequent course. This ensures that students are adequately prepared for their upcoming courses. Therefore, each course is linked to its prerequisite requirements. This aspect is critical as it helps in preventing the creation of semesters with invalid course sequences.

5.2.3. Student Data. The student data, supplied by the university, includes anonymized records of student information. It encompasses details such as the courses each student has taken, transfers, their corresponding grades, the semesters in which the courses were completed, and the degrees earned. This dataset has been incorporated into the PERCEPOLIS server's database, serving as a foundational element for the recommendation approach. In total there are 13968 unique students in the database encompassing 31 undergraduate degrees. Additionally, it is important to note that the students within the database have completed their degrees, meaning they have all the degree requirements for their specific degree.

5.3. TEST CASES

A variety of test cases were employed during the validation process to rigorously assess the proposed scheme. This includes students from different disciplines at different stages of their academic careers with different level of past performance (e.g., GPA). It is important to note that in each test cases there are solid and dashed courses. Solid-course semesters are ones taken by a student while dashed-course semesters are ones being recommended by the algorithm. Additionally, some test cases assume the student passed Math 1140 and Math 1160 prior to starting their academic career.

Each test case offers a range of credit hours for selection. For example, if the range is 12-18, the algorithm can choose semesters that fall within this range. This serves as an extra input, allowing a student to make a choice of the credit hour range they prefer for their generated path.

5.3.1. Test Case 1. Test Case 1, as represented by Figure 5.1, demonstrates the algorithm's capability to construct a graduation pathway for a freshman Computer Science student with 18 total hours. Additionally, the desired credit load input was at 12-18 hours per semester. A single semester is the lowest number of semesters required to run the algorithm to get meaningful results. This is due to the necessity of similar students to generate states, which is not possible without at least a semester of data on which to run the similarity filters and metrics. If there didn't exist a single semester, it would use the set of all students in the same degree program to generate the results. The system is also designed to suggest a course of action that maximizes the student's future GPA. In every semester, it strategically selects courses that not only aim to enhance the GPA for that term but also considers the credit hour threshold that, according to the algorithm's Q-table, promises the greatest future benefit. While it can generate a path as such, it is limited in its future predictions. Since there is a limited number of semesters to compare to, the later predictions become derivative predictions of the initial ones in which accuracy is affected. However, in the end, the algorithm recommended a pathway in which it takes 130 total credit hours to graduate, including 5 credit hours of Math 1160 and 1140.

5.3.2. Test Case 2.1. Test Case 2.1, as represented by Figure 5.2, explores the algorithm's ability to map out a graduation pathway for a junior computer science student with a 4.0 GPA and 70 credit hours by their fourth semester and a credit load of 12-18 credit hours. The case illustrates a graduation path culminating a total of 129 credit hours and a final predicted cumulative GPA of 3.93. Additionally, the algorithm strives to determine the semester credit loads that yield the highest GPA. In this scenario, it was discovered that taking 18 credit hours, followed by 15 and 15 for semesters 5-7, provides the greatest benefit for the student. It is important to note that the final semester is only 6 credit hours. Since the algorithm doesn't optimize time to degree if there are a couple of requirements left but not enough to enable a full-time enrollment, it will just recommend the necessary courses required for graduation. This is an issue that only occurs during the final semester. The algorithm also predicted a GPA of 3.85 for the student in the recommended semesters. Since the course predictions are based on both the student's performance and that of similar students, this information is invaluable for advising. Access to predicted GPA for courses can help identify where a student may need to invest more effort.

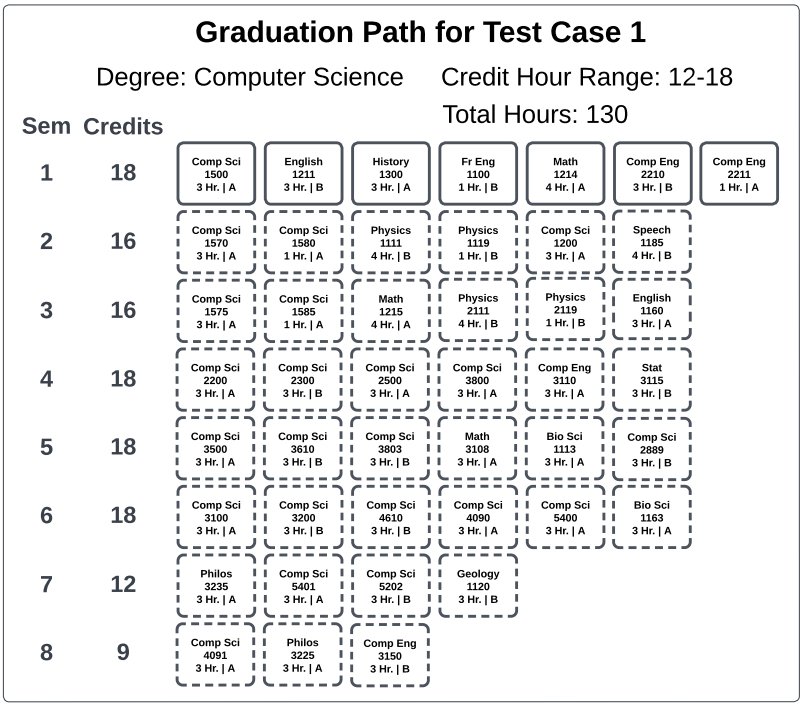

Figure 5.1. Test Case 1

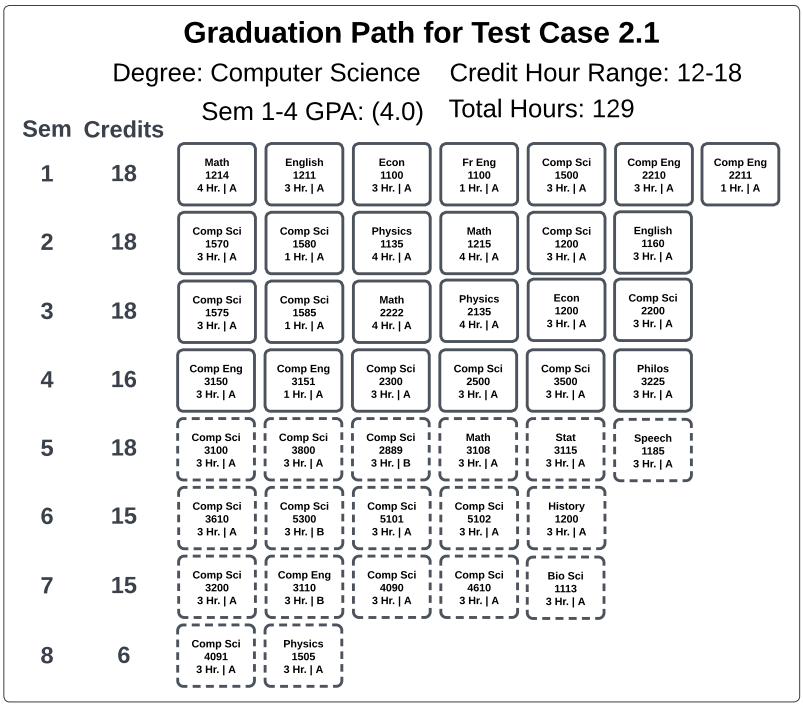

Figure 5.2. Test Case 2.1

5.3.3. Test Case 2.2. Test Case 2.2, as represented by Figure 5.3, serves as an extension of Test Case 2.1, facilitating a comparison between recommendations for academically strong students and those with lower academic achievement. This student, similar to 2.1, has 70 hours by the fourth semester, however, they had a 2.8 GPA within that time. The case shows a graduation path in which there are 128 credit hours and a final predicted GPA of 2.98. This comparison reveals several key points. Initially, there is a commonality in the recommended courses for the first semester, despite differences in expected performance outcomes. This similarity arises from the static nature of course prerequisites, which require a set of foundational courses for further academic progress. However, recommendations start to differ in later semesters, as the basic requirements are met in the earlier semesters. A significant variation is observed in the credit load for the second to last semester of Test Case 2.2, where only 14 credit hours are recommended, as opposed to 15. This adjustment, guided by Q-learning algorithms, indicates a preference for a 14-credit hour semester when a 15-credit option is not viable. Additionally, there is a divergence in the elective courses for degree requirements.

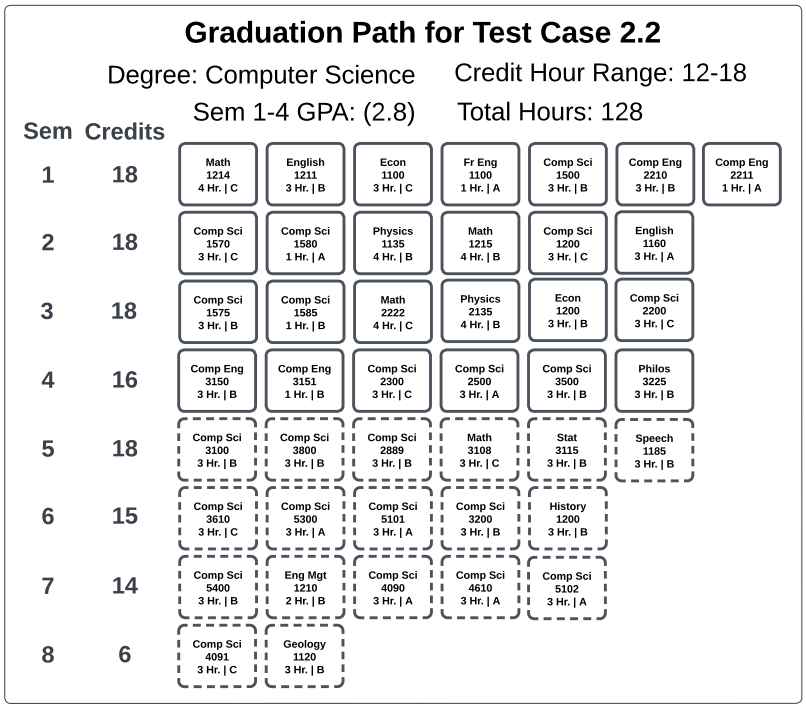

Figure 5.3. Test Case 2.2

5.3.4. Test Case 3.1. Test case 3.1, as represented by Figure 5.4, is presented to illustrate the differences between a student's actual course selections and the path generated for graduation along with the academic predictions. For clarity, test case 3.1 is an actual student's path to graduation taken from the database. Although it is impossible to retroactively alter the student's course choices, this comparison offers valuable insights into how alternative recommendations might have influenced the student's academic performance and trajectory toward graduation. This particular student, drawn from an anonymous database, obtained a 3.09 grade point average throughout their academic journey up to the fourth semester, which is where the comparison with case 3.2 will be made. It's noteworthy to mention that the student received a D grade in Math 1214 during the second semester; however, for prerequisite courses, a minimum grade of C is required before taking select courses. The student was able to take the same course again and get a grade replacement. Another important aspect is the student enrolled in summer courses as represented by 0.5, and the final summer semester dedicated to completing the Physics 2 course, a requirement for graduation. Additionally, summer semesters don't require the minimum of 12 hours. In total, the student completed 128 credit hours to fulfill all graduation prerequisites. They completed semesters 5 through 8.5 with a GPA of 3.53 and ended with a GPA of 3.22.

5.3.5. Test Case 3.2. Test case 3.2, as represented by Figure 5.5, presents a hypothetical scenario in which a student runs the algorithm during their fifth semester. This comparison with 3.1's actual graduation path involves several aspects, including predicted GPA, courses taken, semester loads, and time to degree completion. The algorithm predicts a GPA of 3.38 for the sixth through ninth semesters, slightly lower than the actual student's GPA. However, the prediction's accuracy improves with additional data. The student's GPA before the fifth semester was 3.09, compared to 3.53 afterwards, leading to a lower predicted GPA. There are notable differences in the courses taken, especially in elective requirements offering flexibility in selection, such as Theater, Business, Chemistry, and Engineering Management. The algorithm aims to select options that maximize GPA and future academic success, hence the differences. The recommended credit loads also varied, with the actual student taking 12, 14, 15, and 18 credit hours, while the algorithm suggested within a range of 12-15 credit hours, which it found to be more successful among similar students. Another distinction is the time to degree completion; the student required an extra summer semester, completing their degree in the eleventh semester, whereas the algorithm's path would complete requirements by the ninth full semester. However, it's important to note that this algorithm does not aim to reduce the time to degree. The algorithm prioritizes selecting a current state based on the highest GPA. Consequently,

maximizing the number of credits per semester and selecting prerequisites concerning a paths end is not a priority, and therefore, the time to degree completion is not a primary concern nor does it currently incorporate summer semester selections.

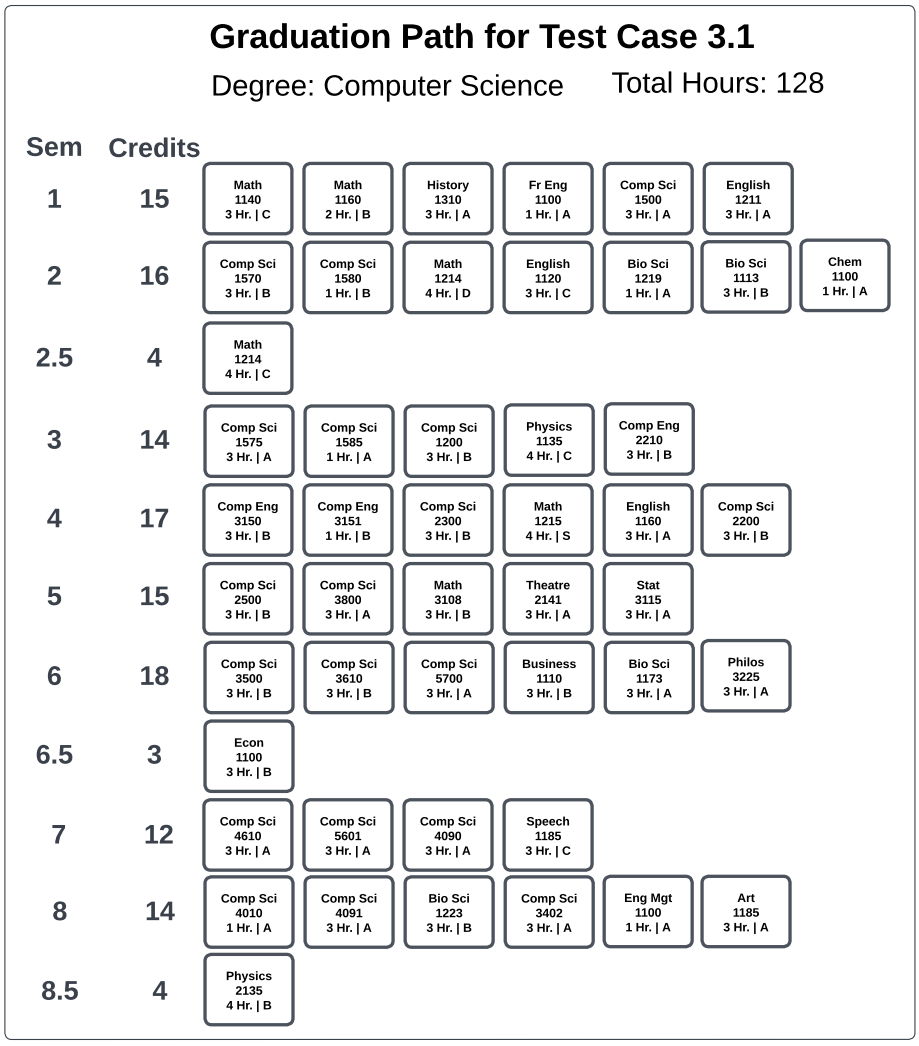

Figure 5.4. Test Case 3.1

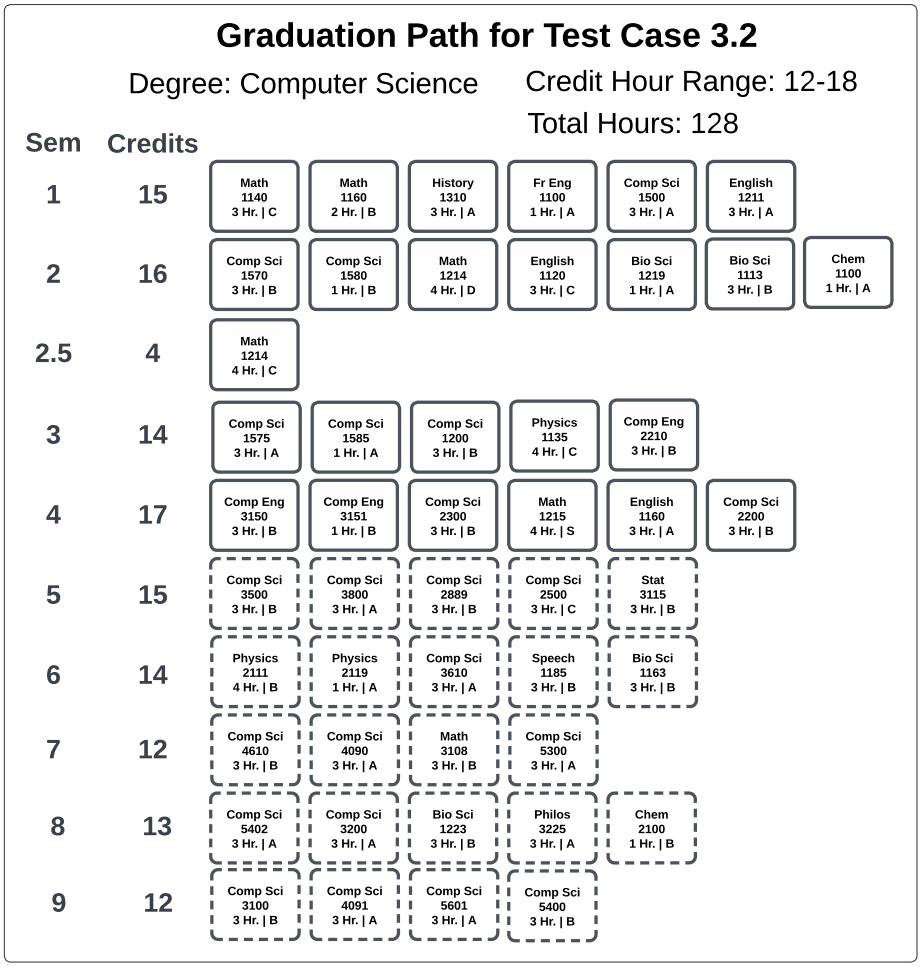

Figure 5.5. Test Case 3.2

5.3.6. Test Case 4.1. Test case 4.1, as represented by Figure 5.6, provides an example of how the algorithm recommends a graduation path for a student in Computer Engineering program. Test cases 4.1 and 4.2 show an example of a student who ran the path generation during their fourth semester with a total of 62 credit hours and a GPA of 2.32. It outlines a pathway that allows the student to complete their degree in 9 semesters, totaling 136 credit hours, and achieve a final cumulative GPA of 2.51. The algorithm aims to identify the optimal semester course loads to maximize student success. It determined that taking between 12 to 16 credits per semester would result in the highest GPA. Additionally, the algorithm calculates predicted grades based on the student's performance and that of similar students. This means it can identify courses that may be

more challenging for a student, as illustrated in test case 2.1, where instead of all A's, select courses were B indicating an increased level of difficultly among students. Similarly, it can also pinpoint courses that generally yield better results for most students and may require less effort than others.

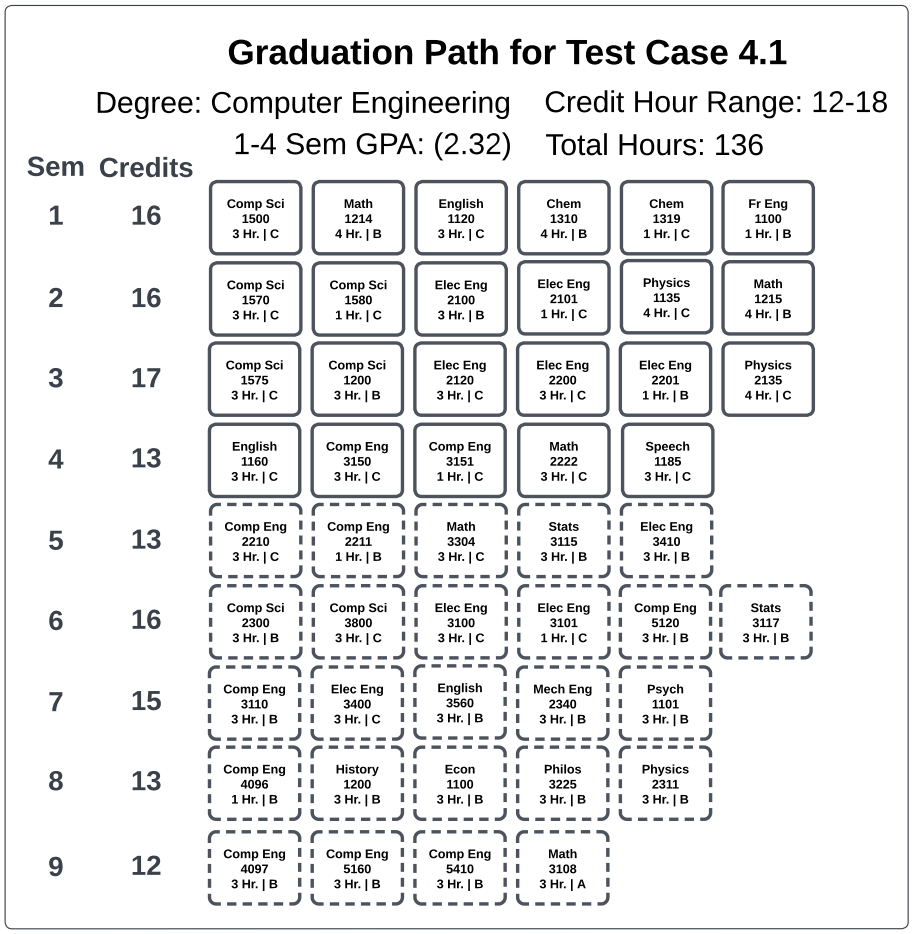

Figure 5.6. Test Case 4.1

5.3.7. Test Case 4.2. Test case 4.2, as represented by Figure 5.7, complements 4.1 by generating a path using the same courses and degree in Computer Engineering but assuming a more academically successful student with a GPA of 3.65. This outlines a pathway through which a more successful student can further enhance their academic performance. The algorithm successfully

created a plan allowing the student to graduate in 9 semesters with a total of 135 credit hours and a final GPA of 3.39. Notable differences emerge in the later semesters, where the conditions become more restrictive, leading to divergent paths. Another distinction lies in the elective courses recommended; the more academically inclined student was advised to take courses in music and art, as opposed to psychology and physics.

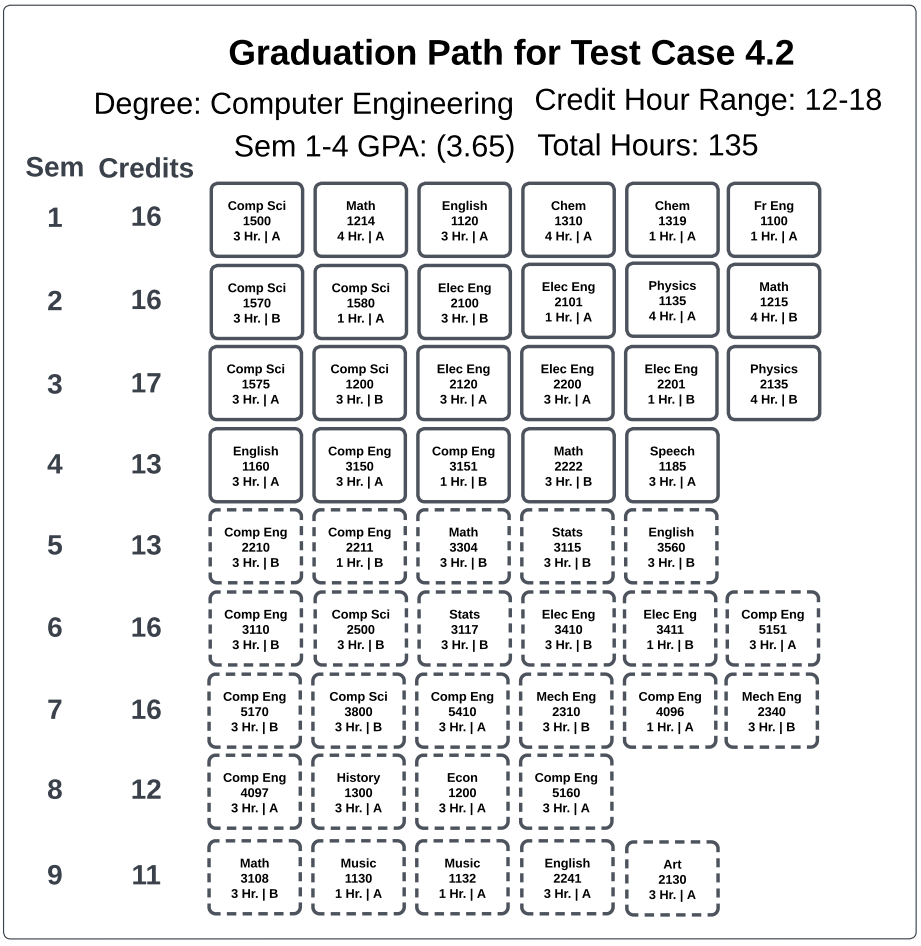

Figure 5.7. Test Case 4.2

5.3.8. Test Case 5.1. Test case 5.1, as represented by Figure 5.8, serves as another example to evaluate the algorithm's prediction performance and path generation capabilities in comparison to a graduated student from the database. Test case 5.1 is the actual students graduation path taken from the student database. This student demonstrated stellar academic achievement by completing their Computer Engineering degree with a cumulative GPA of 3.88. Since this comparison will assume the student runs the path generation after their seventh semester it should be noted that the student achieved a 3.88 GPA for semesters 0.5-4.5 and 5-7.5. Furthermore, the student managed to complete their degree with a total of 130 credit hours. It is also important to make clear a few semesters. It can be seen that the student has English 1120 in their first semester, this can be thought of as the student transferring this credit in before their freshman year. Additionally, it can be assumed the student passed both Math 1140 and 1160 as other students have prior to starting their freshman year. It is also important to note the 4.5 and 7.5 semesters which are both summer semesters which are allowed to be under 12 credit hours.

5.3.9. Test Case 5.2. Test case 5.2, as represented by Figure 5.9, serves as a complement to 5.1, showcasing the pathway generated for the student from 5.1 following their seventh semester. This case spans several key areas. Notably, the similarity between the academic predictions and actual outcomes is remarkable. Both the actual student and the generated pathway boasted a GPA of 3.88 after the 4.5 semester. Furthermore, the recommended pathway concluded by the 8th semester with a total of 135 credit hours. Despite the high accuracy of this test case in mirroring the actual student's journey, it did not prioritize reducing the time to degree completion nor allow summer semester, resulting in an 8th full semester to finish. However, the primary goal of this algorithm is not to hasten graduation but to identify a successful pathway based on the experiences of similar preceding students. It's also worth noting an additional course was taken in the seventh semester, identified by red; although not fulfilling any specific course requirement, it was necessary to maintain full-time student status (12 credit hours), which the algorithm will maintain for all semesters except the last.

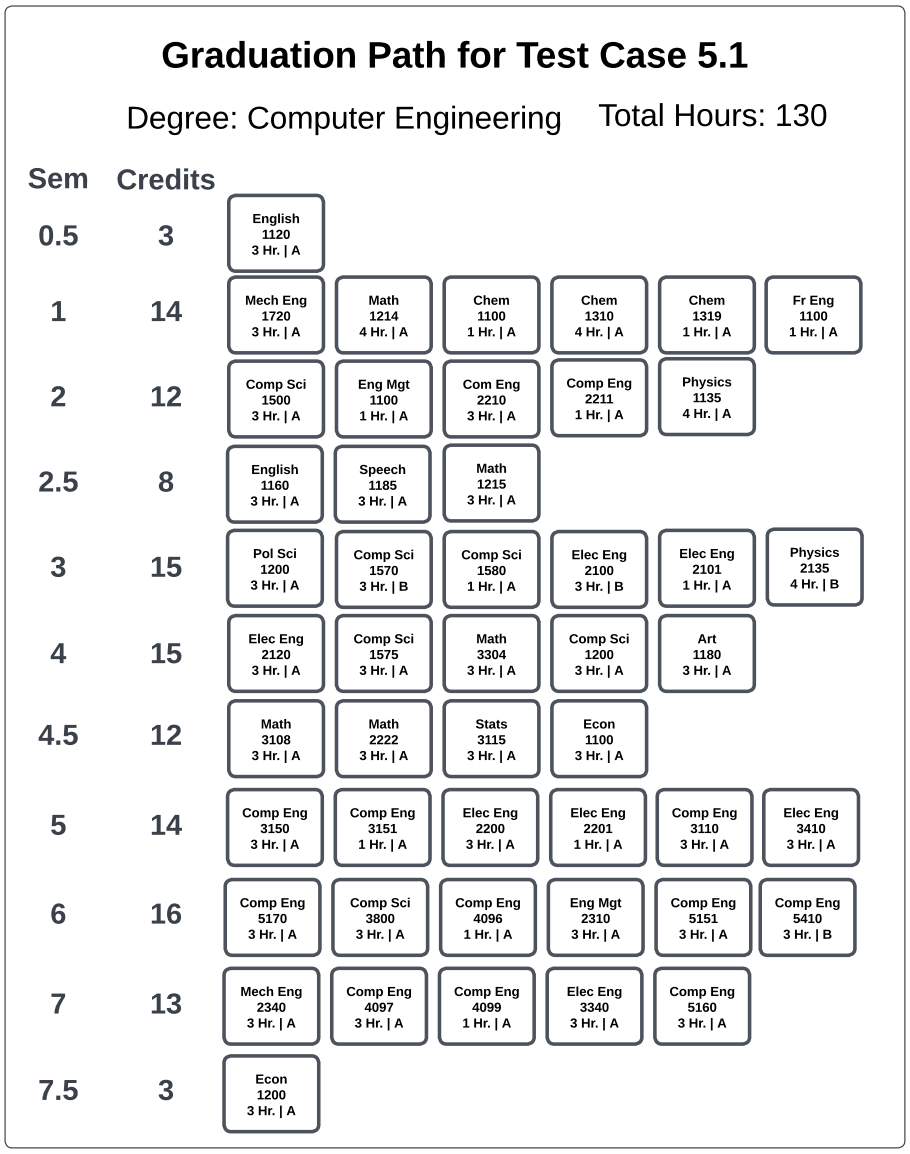

Figure 5.8. Test Case 5.1

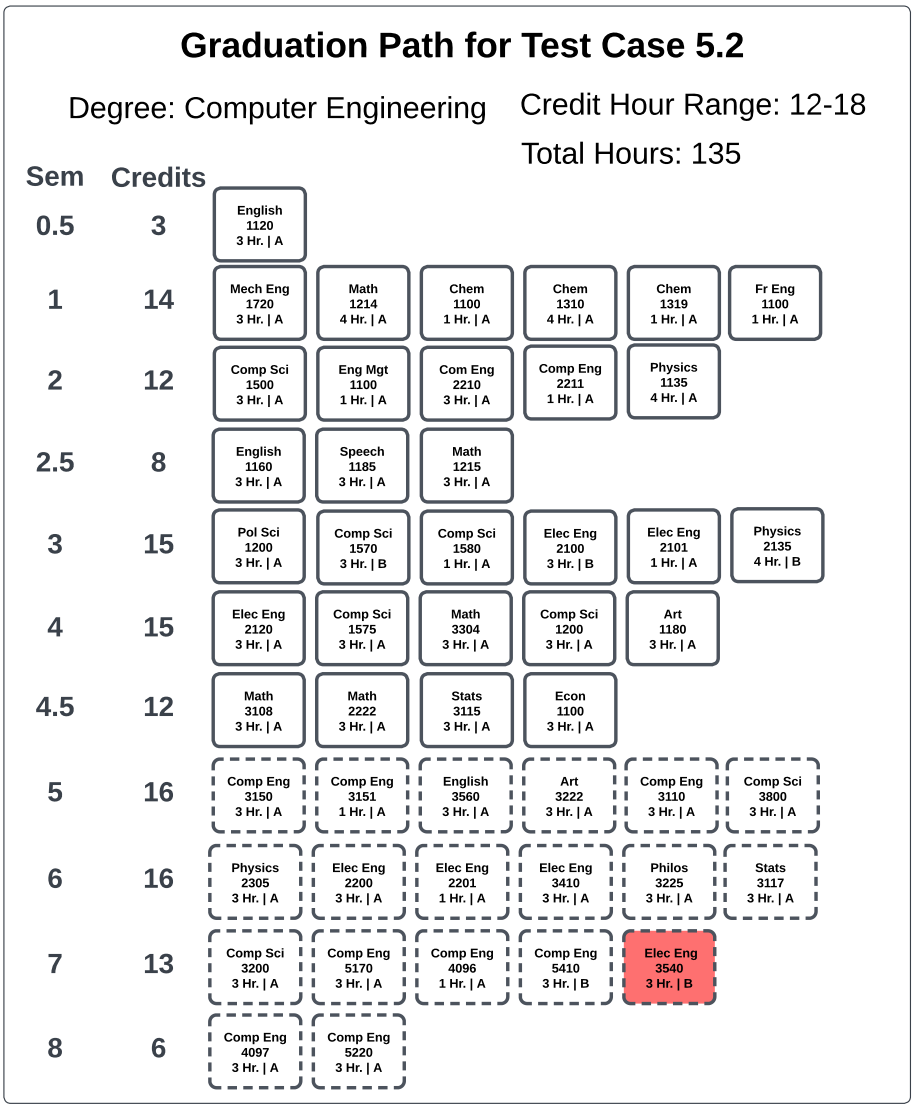

Figure 5.9. Test Case 5.2

5.4. SUMMARY

The results provide significant insight into the potential benefits of this method. The primary goal was to develop a system that, based on past student performance, could recommend semester-by-semester graduation paths conducive to student success. Although this method has not been empirically tested to assess student performance in recommended classes, it can be inferred that the

expected grades and overall student performance would improve following these recommendations. This inference is based on the assumption that students with similar past performances will continue to perform similarly in the future.

One notable finding is the longer duration to degree completion. The number of semesters needed to fulfill all degree requirements ranged from 7 to 8 semesters, exceeding the optimal duration of 7 semesters in some cases, assuming an upper credit limit of 18. This outcome can be attributed to several factors. First, the method prioritizes paths that have proven successful for previous students over optimizing the time to degree. Consequently, selecting the maximum number of credit hours for each semester to reduce time to degree is not prioritized. Secondly, prerequisite constraints play a significant role. For example, in the degrees analyzed, students must complete courses like CS 4096 and CS 4097, which are senior design courses requiring senior standing and specific prerequisites. If these prerequisites are not met in an optimal sequence, it may lead to an additional semester.

Another insight relates to the selection of credit hours. By employing Q-learning, this method identifies the ranges of credit hours that lead to the highest future rewards, thereby emphasizing the effect of a specific credit load on future outcomes. Observations show that although the range spans from 12 to 18 credit hours, the algorithm frequently opts for semesters comprising 15 to 18 credit hours. This indicates that a marginally increased workload could potentially enhance future performance. Furthermore, it's fascinating to observe the influence of GPA performance at the time of assessment on graduation paths. For example, the comparison between a student with a 2.32 GPA and another with a 3.65 GPA in test cases 4.1 and 4.2 reveals differences in their generated paths. While their credit hour limits for the first two semesters are similar, at 13 and 16 respectively, their limits for subsequent semesters and recommended courses diverge. This suggests that students with a specific metrics may not have the optimal semester-load within the similar students and the next best semester-load must then be taken.

In summary, the results are promising and demonstrate the potential benefits that these recommendations could offer students. The method provides insights into how students with similar profiles have successfully navigated their way to graduation. Moreover, by focusing primarily on GPA, it offers a complementary perspective to approaches that prioritize minimizing time to degree.

6. CONCLUSION AND FUTURE DIRECTIONS

The primary objective of this work was to introduce a method that recommends semester-by-semester graduation paths aimed at enhancing student performance. This thesis has successfully presented an innovative approach for generating graduation paths for integration into the PERCEPOLIS system. This method employs a Markov decision process for graduation path generation and utilizes collaborative filtering to identify semesters that have led similar students to academic success. Moreover, it incorporates Q-Learning to determine the most beneficial credit hour loads for a student's future success, as measured by GPA. Overall, this research provides a framework that learns from past data, offering students insightful guidance to plan their education with GPA considerations.

The results of its implementation are highly promising. They demonstrate varied recommendations based on students' academic differences and the types of semesters they can effectively manage. The findings also offer valuable insights into planning graduation paths, suggesting that taking semesters with more credit hours could boost overall academic performance. This method is particularly useful in guiding students from diverse academic backgrounds towards academically advantageous choices. Notably, its effectiveness in delivering accurate and realistic recommendations improves as it accumulates more student data. In essence, this approach equips students with crucial information to aid their journey towards graduation.

Looking forward, several avenues exist for building upon this research. Firstly, this method, primarily aimed at improving student performance, does not address the time to graduation. This oversight has occasionally led to longer graduation times than the "optimal" duration. Future work could therefore focus on leveraging student data to enhance success while simultaneously reducing the time to degree. By analyzing past student achievements, future research could integrate this information into optimizing the balance between academic success and time to graduation.

Secondly, the semester-by-semester generation process offers an opportunity to incorporate student feedback. For example, providing a selection of semester options from which students can choose would allow them to receive tailored recommendations while maintaining control over their graduation trajectory. This would enable a more personalized course selection process, balancing algorithm-driven advice with student preferences.

Finally, future research could explore diverse implementations of the reward mechanism in the Q-learning algorithm. Since academic success might not be the only concern for students, integrating various factors such as interests and personal goals into the reward system could enable more individualized and holistic path generation.

REFERENCES

[1] Morrow, T., Hurson, A. R., and Sedigh Sarvestani, S., 'Algorithmic support for personalized course selection and scheduling,' in 'Proceedings of the 44th IEEE International Computers, Software, and Applications Conference (COMPSAC),' 2020 pp. 143–152, doi: 10.1109/COMPSAC48688.2020.00027.

[2] Dobbins, N., Hurson, A. R., and Sedigh Sarvestani, S., 'Personalizing student graduation paths using expressed student interests,' in '2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC),' 2023 pp. 142–143, doi:10.1109/COMPSAC57700.2023.00027.

[3] Urdaneta-Ponte, M. C., Mendez-Zorrilla, A., and Oleagordia-Ruiz, I., 'Recommendation systems for education: Systematic review,' Electronics, 2021, **10**(14), ISSN 2079-9292, doi:10.3390/electronics10141611.

[4] Basham, J. D., Hall, T. E., Carter, R. A., and Stahl, W. M., 'An operationalized understanding of personalized learning,' Journal of Special Education Technology, September 2016, **31**(3), pp. 126–136, doi:10.1177/0162643416660835.

[5] Javed, U., Shaukat, K., A. Hameed, I., Iqbal, F., Mahboob Alam, T., and Luo, S., 'A Review of Content-Based and Context-Based Recommendation Systems,' International Journal of Emerging Technologies in Learning (iJET), February 2021, **16**(03), p. 274, ISSN 1863-0383, doi:10.3991/ijet.v16i03.18851.

[6] Morsomme, R. and Alferez, S. V., 'Content-based course recommender system for liberal arts education,' in 'Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019),' 2019 pp. 748–753.

[7] Hidayat, A. F., Suwawi, D. D. J., and Laksitowening, K. A., 'Learning content recommendations on personalized learning environment using collaborative filtering method,' in '2020 8th International Conference on Information and Communication Technology (ICoICT),' 2020 pp. 1–6, doi:10.1109/ICoICT49345.2020.9166371.

[8] Klašnja-Milićević, A., Ivanović, M., Vesin, B., and Budimac, Z., 'Enhancing e-learning systems with personalized recommendation based on collaborative tagging techniques,' Applied Intelligence, June 2018, **48**(6), pp. 1519–1535, ISSN 1573-7497, doi:10.1007/s10489-017-1051-8.

[9] Pang, Y., Jin, Y., Zhang, Y., and Zhu, T., 'Collaborative filtering recommendation for mooc application,' Comput. Appl. Eng. Educ., jan 2017, **25**(1), p. 120–128, ISSN 1061-3773, doi:10.1002/cae.21785.

[10] Liu, X., 'A collaborative filtering recommendation algorithm based on the influence sets of e-learning group's behavior,' Cluster Computing, March 2019, **22**(2), pp. 2823–2833, ISSN 1573-7543, doi:10.1007/s10586-017-1560-6.

[11] Li, J. and Ye, Z., 'Course Recommendations in Online Education Based on Collaborative Filtering Recommendation Algorithm,' Complexity, December 2020, **2020**, pp. 1–10, ISSN 1099-0526, 1076-2787, doi:10.1155/2020/6619249.

[12] Rivera, J. E. H., 'A hybrid recommender system to enrollment for elective subjects in engineering students using classification algorithms,' International Journal of Advanced Computer Science and Applications, 2020, **11**(7), doi:10.14569/IJACSA.2020.0110752.

[13] Xu, G., Jia, G., Shi, L., and Zhang, Z., 'Personalized Course Recommendation System Fusing with Knowledge Graph and Collaborative Filtering,' Computational Intelligence and Neuroscience, September 2021, **2021**, pp. 1–8, ISSN 1687-5273, 1687-5265, doi:10.1155/2021/9590502.

[14] Gulzar, Z., Leema, A. A., and Deepak, G., 'Pcrs: Personalized course recommender system based on hybrid approach,' Procedia Computer Science, 2018, **125**, pp. 518–524, ISSN 1877-0509, doi:https://doi.org/10.1016/j.procs.2017.12.067, the 6th International Conference on Smart Computing and Communications.

[15] 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 'Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education,' ITiCSE 2018 Companion, Association for Computing Machinery, New York, NY, USA, ISBN 9781450362238, 2018 p. 175–199, doi: 10.1145/3293881.3295783.

[16] Al-Shehri, H., Al-Qarni, A., Al-Saati, L., Batoaq, A., Badukhen, H., Alrashed, S., Alhiyafi, J., and Olatunji, S. O., 'Student performance prediction using support vector machine and k-nearest neighbor,' in '2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE),' 2017 pp. 1–4, doi:10.1109/CCECE.2017.7946847.

[17] Buniyamin, N., bin Mat, U., and Arshad, P. M., 'Educational data mining for prediction and classification of engineering students achievement,' 2015 IEEE 7th International Conference on Engineering Education (ICEED), 2015, pp. 49–53.

[18] Anand, V. K., Rahiman, S. K. A., Ben George, E., and Huda, A. S., 'Recursive clustering technique for students' performance evaluation in programming courses,' in '2018 Majan International Conference (MIC),' 2018 pp. 1–5, doi:10.1109/MINTC.2018.8363153.

[19] Bagunaid, W., Chilamkurti, N., and Veeraraghavan, P., 'Aisar: Artificial intelligence-based student assessment and recommendation system for e-learning in big data,' Sustainability, 2022, **14**(17), ISSN 2071-1050, doi:10.3390/su141710551.

[20] Alharbi, Z., Cornford, J., Dolder, L., and De La Iglesia, B., 'Using data mining techniques to predict students at risk of poor performance,' in '2016 SAI Computing Conference (SAI),' 2016 pp. 523–531, doi:10.1109/SAI.2016.7556030.

[21] Yağcı, M., 'Educational data mining: prediction of students' academic performance using machine learning algorithms,' Smart Learning Environments, March 2022, **9**(1), p. 11, ISSN 2196-7091, doi:10.1186/s40561-022-00192-z.

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

[23] Chang, P.-C., Lin, C.-H., and Chen, M.-H., 'A Hybrid Course Recommendation System by Integrating Collaborative Filtering and Artificial Immune Systems,' Algorithms, July 2016, **9**(3), p. 47, ISSN 1999-4893, doi:10.3390/a9030047.

[24] Premalatha, M. and Viswanathan, V., 'Course sequence recommendation with course difficulty index using subset sum approximation algorithms,' Cybernetics and Information Technologies, 3919, **19**(3), pp. 25–44, doi:doi:10.2478/cait-2019-0024.

[25] Xu, J., Xing, T., and van der Schaar, M., 'Personalized course sequence recommendations,' IEEE Transactions on Signal Processing, 2016, **64**(20), pp. 5340–5352, doi:10.1109/TSP.2016.2595495.

[26] Morrow, T., 'Personalizing education with algorithmic course selection,' 2017.

VITA

Colton Michael Walker graduated with a Bachelor of Science in Computer Engineering in May 2022 and a Bachelor of Science in Computer Science in December 2022, both from the Missouri University of Science and Technology. Throughout his undergraduate studies, he engaged in research under the mentorship of Dr. Sahra Sedigh Sarvestani and Dr. Alireza Hurson, contributing to the PERCEPOLIS project. As a member of the Honors Academy, he presented his Honors Thesis during his senior year. In 2022, he also gained practical experience as a Web Development Intern at Cboe Global Markets. Then in 2023, he served as both a Graduate Research Assistant and a Graduate Teaching Assistant. He continued his academic journey at the same institution, earning a Master of Science in Computer Science in May 2024.