Algorithmic Support for Personalized Course Selection and Scheduling

Tyler Morrow
Sandia National Laboratories
Albuquerque, USA
Email: tmorrow@sandia.gov

Ali R. Hurson and Sahra Sedigh Sarvestani
Missouri University of Science and Technology
Rolla, USA
Email: {hurson,sedighs}@mst.edu

***Abstract*—The work presented in this paper demonstrates the use of context-aware recommendation to facilitate personalized education, by assisting students in selecting courses and course content and mapping a trajectory to graduation. The recommendation algorithm considers a student's profile and their program's curricular requirements in generating a schedule of courses, while aiming to reduce attributes such as cost and time-to-degree. The resulting optimization problem is solved using integer linear programming and graph-based heuristics. The course selection algorithm has been developed for the Pervasive Cyberinfrastructure for Personalized eLearning and Instructional Support (PERCEPOLIS), which can assist or supplement the degree planning actions of an academic advisor, with assurance that recommended selections are always valid.**

***Index Terms*—personalized education, context-aware recommendation, integer linear programming, PERCEPOLIS**

I. Introduction

The past decades have witnessed efforts towards leveraging technology, in particular databases, pervasive computing, and computational intelligence towards enhancing educational experiences and improving learning outcomes [1], [2], [3], [4], [5], [6].

This paper presents the schedule recommendation aspect of the Pervasive Cyberinfrastructure for Personalized eLearning and Instructional Support (PERCEPOLIS), an educational platform that leverages technological advances - in particular pervasive computing - to facilitate personalized learning, while supporting a networked curricular model [7], [8]. It is cyberinfrastructure in the form of middleware that links databases and learning management systems currently in use at academic institutions to facilitate incremental adoption of the networked and modular curricular model at low cost. The ability to connect tools already in use, without requiring customized computing resources, enables instructors at different academic institutions to collaborate on research and education.

The first step toward the development of PERCEPOLIS has been to understand entities within the educational space; i.e., students, instructors, courses, and technology, and interaction among these entities. To this end, this research has developed an ontology denoted as the modular course hierarchy (MCH), which defines and classifies entities, e.g., institutions, curricula, courses, according to their attributes and relationships with other entities. By arranging these entities into layers based on their classification, a hierarchical structure is formed. A curriculum is comprised of many courses; and courses are comprised of many topics, hence the curriculum can be observed as an entity exhibiting modularity and hierarchical attributes. Modularity is applied to nearly all levels within the MCH and is the primary concept used to define the relationships between entities. The ontology serves as the foundation of the database that underlies PERCEPOLIS and provides the course selection and scheduling algorithms with a meaningful vessel from which to obtain data. PERCEPOLIS can be used to represent curricula (managing entities and relationships) and make context-base recommendations to both students and instructors, who are identified as users within the application. The most recent implementation of PERCEPOLIS is a web-based application that utilizes the MCH in relational database form.

The work described in this article is centered specifically on the recommendation aspect of PERCEPOLIS, which allows identification of a personalized course trajectory based on the individual profile of a student and curricular requirements of their degree program. Figure 1 illustrates this aspect. Section II discusses foundational concepts. PERCEPOLIS's course selection algorithm is presented in Section III and illustrated through an example in Section IV. The article concludes with some final thoughts and ideas for future research.

II. Background

Development and implementation of the proposed course selection methodology, and more broadly, PERCEPOLIS, is founded on knowledge from several areas including: ontologies, data heterogeneity, and semantic similarity. The proposed approach formulates course scheduling as an optimization problem that is solved with linear programming.

An *ontology*, which can be considered a meta-data schema [9], establishes the essential components of that which is being modeled, and its semantics are explicitly defined and machine-processable. By defining shared and common domain theories, ontologies help people and machines to communicate concisely, supporting exchange of semantics, not just syntax [10]. In PERCEPOLIS, ontologies enable the unification of entities and relationships within the educational space into one formal definition and ultimately, allow the course selection algorithm (CSA) presented in this article to operate on educational entities from multiple institutions.

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.00027

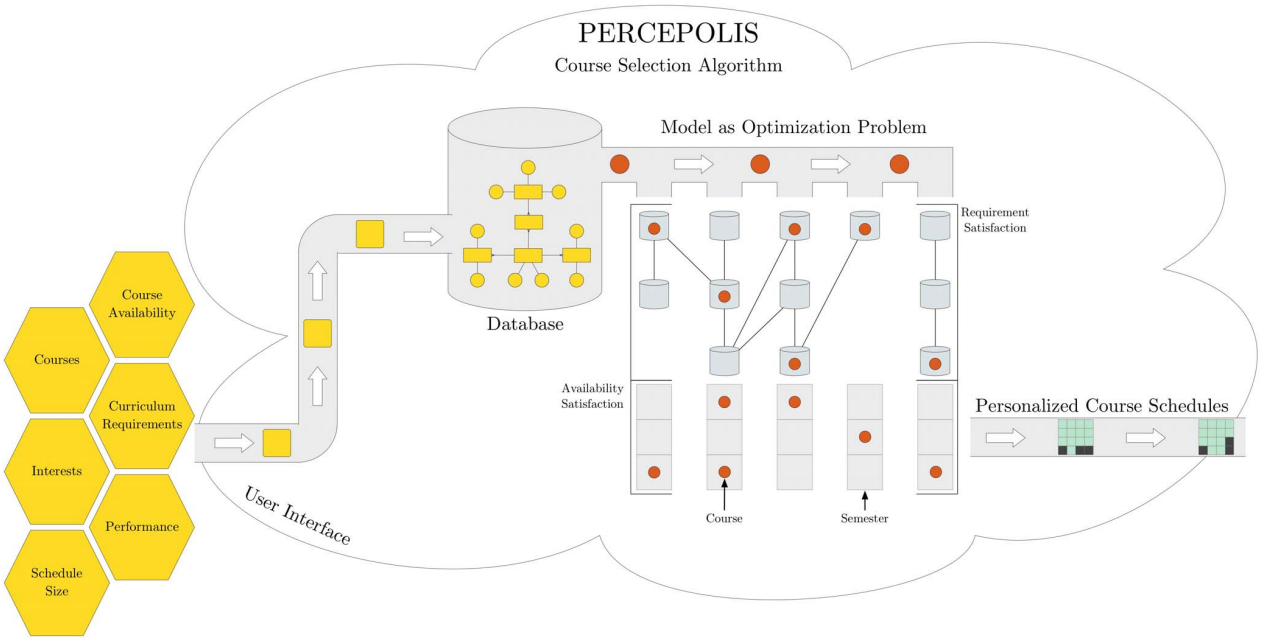

Fig. 1. Recommendation of a personalized course trajectory.

Data heterogeneity is a fundamental challenge associated with merging data from different institutions. An increase in the number of institutions using learning technology results in a wealth of institution-specific data that is distributed and heterogeneous in nature. Each institution maintains its own database(s) of student records, courses, instructors, etc., the autonomy of which must be preserved. At this stage, students and their institutions must provide PERCEPOLIS with their information. Connecting to and retrieving data from the proprietary schemas of institution databases, while technologically possible and vastly more convenient, is not within the current scope of this work. Such information reveals that the same course at different institutions may have slightly dissimilar names and content, which presents a challenge when comparing students at different institutions based on their enrollment behavior. The closely associated challenges of resolving data heterogeneity and identifying *semantic similarity* are the topic of decades of research [11].

A *linear program* (LP) involves the minimization or maximization of a linear function with respect to a set of linear equalities or linear inequalities. Given a set of real numbers $a_1, a_2, \ldots, a_n$ and a set of variables $x_1, x_2, \ldots, x_n$, the general LP has a linear *objective function* of the following form:

$$f(x_1, x_2, \ldots, x_n) = a_1x_1 + a_2x_2 + \cdots + a_nx_n = \sum_{j=1}^{n} a_jx_j.$$

It is the objective function for which the minimum or maximum value is sought, and in practice constraints are applied which limit possible values. These constraints are represented as linear equalities or inequalities depending on the form being used. The problem becomes an integer linear program (ILP) when the variables $x_1, x_2, \ldots, x_n$ may take only integer values. The most common methods used to solve linear programs are simplex-based. ILPs can be solved with tools such as those provided in the Computational Infrastructure for Operations Research (COIN-OR) [12]. This research considers course selection to consist of several different linear programming problems, examples of which are requirement satisfaction and semester satisfaction. In the case of requirement satisfaction, the goal is to minimize the number of credit hours while ensuring that the right courses are used. For semester satisfaction, the goal is to keeping credit hours within some range while preserving requisite constraints.

III. Methodology

This section is intended to address and discuss key conceptual underpinnings of the course selection algorithm (CSA) used by PERCEPOLIS to generate personalized schedules for a given student. The CSA consists of four stages, as depicted in Figure 2:

1) Data Collection
2) Schedule Creation
   a) Course Selection
   b) Semester Selection
   c) Module Selection
3) Schedule Enumeration
4) Schedule Presentation

A course selection algorithm (CSA) finds a set of courses for a particular student, grouped and ordered by time, which satisfy the student's curricular requirements. The total adherence to institution constraints is mandatory because with it comes

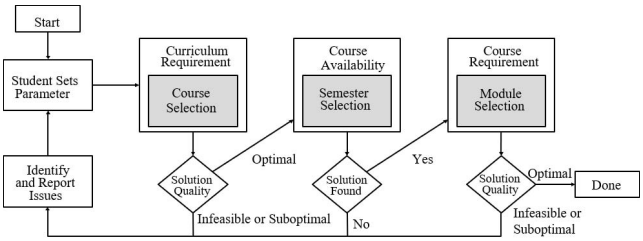

Fig. 2. Flow of operations.

the guarantee of eventual graduation by the student if they are successful. Once validity is established, a course selection and the CSA from which it came can be further judged in numerous ways, such as: level of personalization, time to graduation, difficulty, and probability of student success.

The PERCEPOLIS course selection algorithm addresses the following areas of emphasis (AoEs) for a target student:

1) Requirements (including, but not limited to, those set by an institution);
2) Personal interests;
3) Time restrictions; and
4) Increased likelihood of higher performance.

No claim is made that an optimal solution that addresses all AoEs will be generated. This stems from the nature of some AoEs for which satisfaction can only be shown after the fact by feedback, and even then measures can be interpreted subjectively. For example, consider the objective of increasing the likelihood of higher performance. A practical approach is to use heuristics derived from the known performance of similar students. With this information, courses could be selected in such a manner that students having all backgrounds are more likely to have a higher GPA. The focus is therefore on selections which are optimal in some ways while "good enough" in others as determined by certain thresholds for AoE satisfaction. A selection can be optimal in terms of time-to-degree by not allowing the student take any more than the minimum number of courses needed to graduate, as determined from the institution's rules. A selection can also be optimal in terms of personal interests by catering to all of the student's interests. However, a selection does not completely guarantee that selecting one course over another will result in a higher GPA for the target student, and so it is this probability which introduces partial sub-optimality. The CSA sets conditions of satisfaction for each AoE, based on the information provided and then attempts to create a schedule that satisfies all of them.

The next subsection addresses the types of the data on which the CSA operates. This data is then analyzed to derive the decision variables, constraints, and objective function to treat this problem as an ILP.

A. Data

The PERCEPOLIS CSA generates recommended course schedules with consideration of four AoEs: degree requirements, interests, increased likelihood of higher performance, and desired time-to-degree. The following list outlines all of entities and their properties in order to perform the algorithms needed to carry out course selection.

- Student
  - id: unique identifier
  - $name$: the student's first and last name
  - $interests$: a list of keyword interests
  - $curricula$: a list of curriculum entities
  - $completedCourses$: a list of completed courses with known grade and semester
  - $explicitCourses$: a list of explicitly desired course entities provided by the student
  - $implicitCourses$: a list of implicitly desired course entities inferred from $interests$ [1]
  - $explicitModules$: a list of explicitly desired module entities provided by the student
  - $implicitModules$: a list of implicitly desired module entities inferred from $interests$ [2]
  - $completedModules$: a list of completed modules with known grade
- Curriculum
  - id: a unique identifier
  - $name$: a label describing the nature of the curriculum
  - $requirements$: a list of requirement entities needed to complete the curriculum
- Requirement
  - id: a unique identifier
  - $name$: a label describing the nature of the curriculum
  - $creditThreshold$: a numeric threshold towards which courses are applied and above which the requirement is considered fulfilled
  - $courses$: a list of courses that can be used to complete the requirement
- Course
  - id: a unique identifier
  - $name$: a label describing the nature of the curriculum
  - $credits$: a numeric value that can applied towards one or more requirement credit thresholds
  - $creditThreshold$: a numeric threshold towards which modules are applied and above which the course is considered fulfilled
  - $creditsRequired$: a numeric threshold representing the number of credit hours a student must have completed in previous semesters before taking this course
  - $modules$: a list of modules that can be used to complete the course
  - $prerequisites$: a list of prerequisite courses
  - $postrequisites$: a list of postrequisite courses [3]
  - $corequisites$: a list of corequisite courses
  - $consequisites$: a list of consequisite courses [4]
  - $availabilities$: a list of semesters in which the course is known to be available
  - $unvailabilities$: a list of semesters in which the course is known to be unavailable
  - $opportunity$: the number of courses to which this course leads

[1]Note that interest keywords are assumed to have already been used to populate the sets of implicitly desired courses and modules.

[2]See footnote 1

[3]Courses for which this course is a prerequisite

[4]Courses that have to be taken consecutively

- Module
  - id: a unique identifier
  - $name$: a label describing the nature of the curriculum
  - $credits$: a numeric value that can applied to a course's $creditThreshold$
  - $prerequisites$: a list of prerequisite modules
  - $postrequisites$: a list of postrequisite modules
- Semester
  - id: a unique identifier
  - $name$: a label describing the nature of the curriculum
  - $beginsOn$: a real-world beginning date/time
  - $endsOn$: a real-world end date/times
  - $maxCredits$: a maximum number of credit hours allowable
  - $minCredits$: a minimum number of credit hours allowable

1) Institutional Requirements: For students enrolled in a traditional college or university, institutional requirements are somewhat rigid. In order to produce a valid course schedule, the CSA needs to know what courses are available at a given institution, credit requirements, how those courses relate (prerequisites, corequisites, consequisites), and what courses must be taken in order to complete a given curriculum. If only institutional requirements and student transcripts are known, the CSA produces valid schedules. It is assumed that the CSA's database has up-to-date student transcript information about what courses they have taken. This gives the CSA the ability to use previously taken courses towards curriculum requirements. Previously taken modules can be applied to multiple courses, consequently shortening the amount of time it takes to complete the course. Alternatively, completing a module shared between courses enables the possibility of substituting that time with a more interesting module or more challenging one.

Less formal courses, bear no time restrictions, do not fall under a curriculum in the traditional sense. Online courses, for example, are often available on-demand, so with respect to the CSA this means they can be started at any particular time. Regardless, the CSA requires that time steps be defined for any entity on which availability selection is eventually carried out, for traditional setting, the time steps are obviously semesters or quarters. Discretizing the time within semesters to do availability selection on modules makes little sense, since if a course is offered, the modules within are assumed to be available.

2) Interest: The CSA will try to personalize schedules based on the explicit and implicit interests of the target student. Students may explicitly specify courses and modules they want to take, or if they are unsure, they can provide keywords for topics they want to learn. The CSA will incorporate these words or phrases into the final schedule by attempting to match them to existing courses and modules. Unfortunately, a semantic gap typically exists between the keywords provided by the student and the information available for courses and modules, which may be limited to simply a name. To address this semantic gap, the CSA makes use of the Word2Vec algorithm developed by Google. The Word2Vec algorithm is first trained on a quality, English-text corpus in order to produce meaningful vector representations of words in the language. Numerous corpora for any desired language can be found with a simple Google search. Once all of the word vectors have been generated as a binary file, cosine similarity can be used to find the words which are most similar to some target word.

3) Promoting Improved Performance: The third AoE involves increasing the target student's likelihood of higher performance in a manner which is balanced with the other AoEs. This might be done by eliminating certain courses (where performance is abnormally low) from consideration prior to the course selection stage. Estimating how a target student will do in a course is a delicate and multi-faceted problem. The number of factors which result in a particular grade for any given student are numerous, while the data for most of these factors is difficult to obtain and/or nonexistent. Therefore an approach is proposed that both relies on data which is surely available (course grades) and attempts to reduce some of the noise introduced by unknown factors.

If performance data was available, the PERCEPOLIS would identify a cluster of students to which the target is most similar, where similarity is determined by factors such as; the grades received in mutual courses, having the same major(s), etc. Another requirement for this is that the peer students must be further along in their academic career than the target student, which includes both former and current students of the institution. If a clustering method such as k-means were used, the value of k would need to be set based on the analysis of a real dataset. To avoid this, agglomerative hierarchical clustering is used which starts with all students in their own cluster and iteratively merges clusters which are the closest. Whenever, the target student's cluster is sufficiently large, the clustering can stop and the peer students present in the cluster can provide the basis for estimation. New students with no prior courses present a special case that requires alternative sources of information for comparison including, but not limited to, high school performance and ACT/SAT score. Course recommendations made for improving performance are more important early on in a student's academic career in order to promote retention, and as such, properly making use of alternative information remains an important open issue.

Before clustering, metrics for the distance between students and between clusters must be defined. Since course history is the focus, each student is represented as a vector of all the courses taken by the target student. Let the total number of students in the target's major(s), including the target, be m, and let n be the number of courses taken by the target student. Also let h_i be the vector of length n for the i-th student, and let H be the set of course history vectors containing $h_i \forall i \in [1, m]$. Lastly, let $h_{i,g}$ denote the grade received by student i for course g where $g \in [1, n]$. Deriving a distance measure is pretty easy with this representation, the most common of which would be Euclidean distance. For two student vectors,

the Euclidean distance between them can be found as follows:

$$Euclidean(h_i, h_j) = \Big(\sum_{k=1}^{n} |h_{i,k} - h_{j,k}|^{\frac{1}{2}} \Big)^2 \quad (1)$$

However, other distance metrics such Minkowski, Mahalanobis, City Block, etc. also exist, and even similarity measures such as cosine similarity can be modified to produce a measure of distance. The results of using any particular metric on real data remain to be explored.

The criteria by which clusters of students are merged must also be defined since the metric defined previously is only for computing the distance between two particular students, not clusters of students. This is known as the "linkage criteria." There are many types of linkage criteria, however, our recommended approach is group average linkage. As a point of information, this technically turns the clustering method into what is known as an Unweighted Pair Group Method with Arithmetic Mean (UPGMA). In UPGMA, the distance between two clusters is the average distance between the vectors in each cluster. Given two clusters of student vectors C_1 and C_2, the distance between them would be defined as:

$$\frac{1}{|C_1| \cdot |C_2|} \sum_{a \in C_1} \sum_{b \in C_2} distance(a, b) \quad (2)$$

where $(||)$ represents the length of a vector and $distance(a, b)$ is the chosen distance metric. The results of different linkage criteria should be compared on real data in order to determine the more appropriate method. Nevertheless, we chose the UPGMA approach primarily because it tends to join clusters with small variances first, this can be interpreted as leading the target student to a cluster in which the students are less dispersed and therefore slightly more similar. When the size of the target student's cluster becomes sufficiently large, the peer students in the cluster are extracted. Note that when the vectors were formed, only the courses that were taken by the target student were used. In other words, the clusters were not formed using the entire history of peer students, but rather only the portion of their history which was the same as the target student. With similar peer students obtained, the mean grade for each of the courses not shared with the target student is calculated. The calculated mean grades will be used to eliminate courses with low average grades for curriculum requirements with many course options. The threshold for such removal yet another open question. However, if the target student is a high performer, the threshold should be removed to allow the target student to take courses perceived as more difficult. This prevents the objectionable scenario where harder courses are never selected. Consequently, the aforementioned scenario would only apply to underperforming students by disallowing the selection of courses where the difficulty is perceived to be abnormally high. Future work may consider investigating whether or not clustering can be improved by deriving distance with different measures. Moreover, incorporating consideration for the order in which courses are taken may lead to a more accurate measurement. A starting point for considering the order in which courses were taken would be an edit distance calculation that finds the best global alignment and then subsequently considering grades [49, 48]. It would also be worthwhile to expand this approach to the module level as it would offer a higher level of detail with respect to a student's behavior and performance.

4) Desired Time-to-Degree: The final AoE relates to the amount of time the student would prefer to spend pursuing a degree. Schedule height, or the number of credit hours per semester, is initially set by the target student's institution as follows:

1) $minCr_t$: the minimum number of credit hours for semester $t \in T$, where T is the time-ordered set of all semesters
2) $maxCr_t$: the maximum number of credit hours in semester t

Student provided parameters are defined as follows:

1) $minCr_{s,t}$: the minimum number of credit hours for student $s \in S$ where S is the unordered set of all students, and for semester $t \in T$ where T is the time-ordered set of all semesters
2) $maxCr_{s,t}$: the maximum number of credit hours for student s in semester t
3) $maxSemesters_s$: the maximum number of semesters preferred by student s

The CSA always tries to make the schedule as short as possible independent of the value of $maxSemesters_s$. $maxSemesters_s$ is used by the CSA to determine whether or not a created schedule adheres to the target student's preference. If no value for $maxSemesters_s$ is provided, the CSA will set $maxSemesters_s$ to $scheduleLength$ which is the resultant length of the created schedule, ensuring the schedule is valid with respect to length. Since students cannot override institution limits, the following must hold:

$$minCr_t \leq minCr_{s,t} \leq maxCr_{s,t} \leq maxCr_t \quad (3)$$

Let $totalCredits_t$ be the total credit hours placed in semester t for the target student, and note that its value must fall within the minimum and maximum institution and student parameters. In order for a schedule to be institutionally valid, the following must hold:

$$0 < minCr_t \leq totalCredits_t \leq maxCr_t, \forall t \in T \quad (4)$$

In order for a schedule to be preferentially valid, the following must also hold:

$$0 < minCr_{s,t} \leq totalCredits_t \leq maxCr_{s,t}, \forall t \in T \quad (5)$$

Since the bounds do not have to be uniform for all semesters, a target student could adjust his/her $minCr_{s,t}$ and $maxCr_{s,t}$ such that the workload would change semester-by-semester rather than being uniform. However, if the gap between a minimum and maximum is too small, the CSA could fail to find a preferentially valid solution.

B. Schedule Creation

As shown in Figure 2 the schedule creation process within the CSA consists of the following:

1) Curriculum/Requirement Satisfaction (Course Selection)
2) Course Availability Satisfaction (Semester Selection)
3) Course Satisfaction (Module Selection)

Schedule creation is broken down into a series of three stages, each of which seeks to find an optimal or valid solution to one or more ILPs before passing the results to the next stage. The way in which each ILP is modeled (most notably in constraint derivation for requirements) is what differentiates the CSA from similar works. Refer to figure 2, satisfaction stages 1 and 2 represent ILPs, however, stage 2 is solved using proprietary algorithms.

1) Curriculum requirement satisfaction: Curriculum requirement satisfaction requires finding a selection of courses that satisfy all of the curriculum requirement credit thresholds. A *requirement* entity consists of the courses which can satisfy it and a credit hour threshold to which those courses can be applied. An optimal solution for curriculum requirement satisfaction is a set of courses which minimizes the total number of credit hours taken, while simultaneously adhering to all constraints. To formulate an optimization problem, *decision variables*, *constraints*, and an *objective function* must be created from the data.

Decision variables are created to represent the decision to use a particular course i to satisfy a particular requirement j. Formally, the set of all decision variables for curriculum requirement satisfaction is defined as:

$$X = \{x_{i,j} : x_{i,j} \in \{0,1\}, j \in R, i \in j.courses\} \quad (6)$$

where R is the set of the target student's requirements and $j.courses$ is the set of all courses which can satisfy j. The entire set X is produced directly from the requirements R that must have been previously provided by institutions.

It is certainly possible that a single course may satisfy more than one requirement. The general rule is that a course can only be used towards one requirement within any given curriculum. To enforce this rule, constraints must be created such that decision variables representing the same course within a curriculum are not selected more than once. Therefore, the sum of the decision variables for course i must not exceed 1. Figure 3 is intended to highlight this issue.

A requirement's course list can sometimes include a course as well as one or more of its prerequisites. As such, up to this point, it would be possible to select a course without selecting all of its prerequisites, this is obviously not valid. This is a problem that arises within elective requirements where many options are available. Moreover, it must be the case that if a student is interested in a course, all of the prerequisites for the course must also be added. Therefore, it is entirely possible that too many constraints can lead to solutions which do not minimize the number of credit hours. Similarly, a co-requisite relationship can be ensured in the same way as a prerequisite

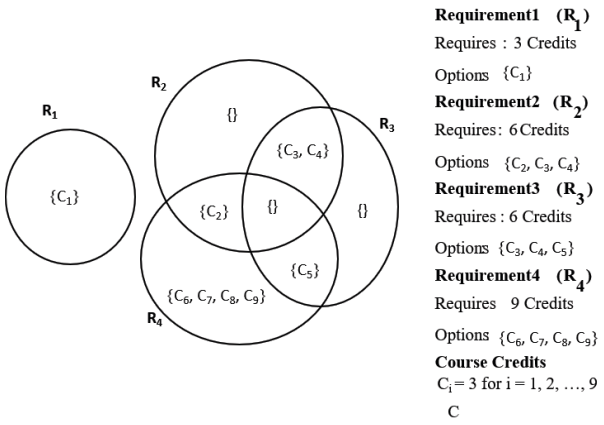

Fig. 3. Overlapping constraints.

relationship since both are, in essence, the constraint that one course cannot be selected without another.

Additionally, a **constraint** must be created for every requirement's credit hour threshold. For a given requirement, its decision variables are multiplied by their respective course credit and then summed. This sum must be greater than or equal to the requirement's credit hour threshold. The greater than aspect of the inequality creates flexibility in the event that requirement credit thresholds are not perfectly divisible by the requirement's courses.

If one or more courses are implicit or explicit interest of the target student, then a constraint is added for each course such that the sum of all of the decision variables for that course are greater than or equal to 1. It is assumed that there is a requirement $r \in R$ for each course of personal interest. However, if the student is interested in a large number of courses, it is very likely that his/her total number of credit hours will not be minimal.

The CSA also adds constraints to ensure the use of every prior course or module taken by the student. Previously created constraints for reusability will ensure that courses are not improperly chosen more than once, while modules do not currently have reusability constraints applied to them. By doing this, previously taken courses and modules are used wherever possible to work towards a goal of shortening the time-to-degree while satisfying curriculum requirements.

A final constraint is also added for the sum of requirement thresholds which provides a lower bound for the eventual **objective function**, $F(x)$ (to be minimized):

$$F(x) = \sum_{x_{i,j} \in X} u_i x_{i,j} \quad (7)$$

where u_i is the credit value for course i. With the objective function, decision variables, and constraints all collected, the ILP is formed and can now be solved. As an ILP, an established method such as COIN-OR branch and cut can be used to find an optimal solution [12]. The result of this

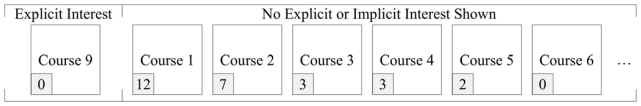

Fig. 4. Prioritized course list

satisfaction stage is a set of courses, C_s, which is guaranteed to satisfy the student's curriculum requirements.

2) Availability satisfaction: Availability satisfaction requires placing each course into the earliest semester possible, in an attempt to shorten the time-to-degree. This is done by iterating over each future semester and assigning a course to the semester if possible. For each semester, the CSA will iterate down a prioritized list of courses and place the courses which are available and have all of their prerequisites met.

The additional properties are added so that the CSA can keep track of which courses have been added to the schedule, preventing courses from being scheduled more than once. The list of courses, C_s, obtained in the previous stage are first grouped by the level of interest the student has in them. Priority is first given to courses of explicit interest, then implicit interest, and finally low interest. Low interest simply represents that the student has not displayed any direct interest. Each interest group is then sorted in descending order of *opportunity value* represented as $c.opportunity$, where $c \in C_s$. To establish opportunity value, a directed acyclic graph is created where the vertices are courses and arcs are established by the prerequisite relationships between courses. Figure 4 depicts the prioritized course list resulting from the prerequisite graph given in Figure 5.

When a student completes a course, the student unlocks the opportunity to take any subsequent courses in future semesters. Each course can be weighted naturally in terms of the number of courses to which it provides access, whether the access be immediate or not. A course is therefore weighted for its direct and indirect post-requisites. In order to ensure higher priority and earlier placement, a course's opportunity value should be greater than the opportunity value of any of its post-requisites. Courses are given their opportunity value by simple graph traversal. An artificial root node could be added to connect the entire graph and serve as a starting point, but that is an implementation decision of little concern here.

The definition of opportunity value for a course is the number of post-requisites plus the sum of opportunity values of the post-requisites. More formally, let the function $O(c)$ representing the opportunity value of course c be defined as:

$$O(c) = \begin{cases} cpl + \sum_{d \in c.postrequisites} O(d), & \text{if } cpl > 0 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where $cpl = c.postrequisites.length$.

Note that the longest path in the prerequisite graph represents the shortest possible number of semesters to graduation, a fact which can serve for validation purposes if desired. Once

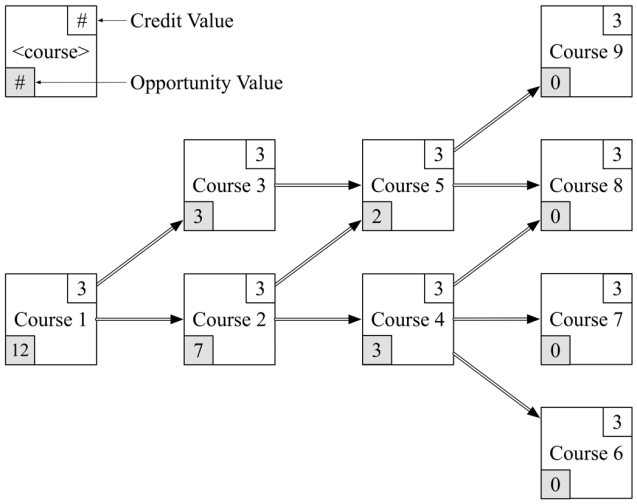

Fig. 5. Prerequisite graph with opportunity values

every course has an opportunity value, the prioritized list of courses can be formed. The result $O(c)$ is shown in Figure 5.

With a prioritized list of courses for the target student, C_p, every course $c \in C_p$ must be placed in some semester $t \in T$. However, before a course can be added to a semester, the CSA must know that it is allowed to do so. First, the number of credit hours earned thus far, a, is compared against the course's $creditsRequired$ to see if enough credit hours have been earned in previous semesters. If not, the function returns $false$. Secondly, the function verifies that all of the course's prerequisites have been taken in prior semesters. Thirdly, the function checks to see if adding the course and its co-requisites will exceed the maximum allowable credit hours for the semester. Finally, if the course is known to be unavailable in the given semester, then it cannot be added.

For each semester and for each course in the prioritized course list, the CSA determines whether or not the course can be added to the semester. If so, the course is added along with all of its co-requisites. Once a course is added, it is locked into that position until all courses are place or there are no more semesters left. The semester limit can be determined by the student or institution, but lower limits can reduce the chance of finding a suitable result.

3) Course Satisfaction: Up to this point, the CSA has dealt with courses, but the final stage involves selecting the modules (or topics) with the selected courses that student will take. This process is a reimagining of the curriculum requirement satisfaction stage where courses are now the entities that must be satisfied. Consequently, decision variables represent the binary decision to use a module for a course. The course credit constraints, prior module constraints, interest constraints, and total credit constraint are all used. However, modules are sometimes marked as required which produces additional constraints ensuring that their respective decision variables are selected. The objective function, minimizing the sum of credit hours, remains the same.

TABLE I
SAMPLE CURRICULUM REQUIREMENTS

Requirement 1	Needs three credit hours
Course Options	1
Requirement 2	Needs six credit hours
Course Options	2,3,4
Requirement 3	Needs six credit hours
Course Options	3,4,5
Requirement 4	Needs nine credit hours
Course Options	2,5,6,7,8,9

IV. RESULTS

The results of this section are produced by selecting a set of courses and modules for a target student, with the additional intentional constraints to achieve different degrees of personalization. The sample student given serves to illustrate both how the algorithm works, as well as the effect that prior courses and known interests have on the selection process.

In this scenario, the goal is to find a course, semester, and module selection for a student who has previously completed course 3 and has a known interest in taking course 9 and all of its modules. Each course is assumed to be thee credit hours. The curriculum requirements for the given student is given in Table I, while the prerequisite graph and prioritized course list for the given student are shown in Figures 4 and 5, respectively.

A. Course selection

As a reminder, curriculum requirement satisfaction deals with making a course selection that fulfills all requirements within a curriculum, includes courses of interest to the student, minimizing the total number of credit hours taken in order to reduce time-to-degree. What follows are the decision variables, constraints, and objective function that describe the course selection problem as an integer linear program.

Decision variables:

$$\begin{aligned} X = \{x_{i,j} : & x_{1,1}, x_{2,2}, x_{3,2}, x_{4,2}, \\ & x_{3,3}, x_{4,3}, x_{5,3}, x_{2,4}, \\ & x_{5,4}, x_{6,4}, x_{7,4}, x_{8,4}, x_{9,4}\} \end{aligned}$$

where i = course, j = requirement.

Requirement reusability constraints:

$$\begin{aligned} x_{2,2} + x_{2,4} &\leq 1 \\ x_{3,2} + x_{3,3} &\leq 1 \\ x_{4,2} + x_{4,3} &\leq 1 \\ x_{5,3} + x_{5,4} &\leq 1 \end{aligned}$$

TABLE II
SOLUTION: COURSE SELECTIONS.

	Solutions					
Requirement	1	2	3	4	5	6
1	1	1	1	1	1	1
2	2,4	2,3	2,3	2,3	2,4	2,4
3	3,5	4,5	4,5	4,5	3,5	3,5
4	6,7,9	6,8,9	7,8,9	6,7,9	7,8,9	6,8,9

Requirement credit constraints:

$$\begin{aligned} C_1 x_{1,1} &\geq R_1 = 1 \\ C_2 x_{2,2} + C_3 x_{3,2} + C_4 x_{4,2} &\geq R_2 = 6 \\ C_3 x_{3,3} + C_4 x_{4,3} + C_5 x_{5,3} &\geq R_3 = 6 \\ C_2 x_{2,4} + C_5 x_{5,4} + C_6 x_{6,4} & \\ + C_7 x_{7,4} + C_8 x_{8,4} + C_9 x_{9,4} &\geq R_4 = 9 \end{aligned}$$

where C_i represents the credit value for course i.

Prerequisite constraints:

$$\begin{aligned} x_{1,1} - x_{2,2} - x_{2,4} &\geq 0 \\ x_{1,1} - x_{3,2} - x_{3,3} &\geq 0 \\ x_{2,2} + x_{2,4} - x_{4,2} - x_{4,3} &\geq 0 \\ x_{2,2} + x_{2,4} - x_{5,3} - x_{5,4} &\geq 0 \\ x_{3,2} + x_{3,3} - x_{5,3} - x_{5,4} &\geq 0 \\ x_{5,3} + x_{5,4} - x_{8,4} &\geq 0 \\ x_{5,3} + x_{5,4} - x_{9,4} &\geq 0 \\ x_{4,2} + x_{4,3} - x_{7,4} &\geq 0 \\ x_{4,2} + x_{4,3} - x_{8,4} &\geq 0 \\ x_{4,2} + x_{4,3} - x_{6,4} &\geq 0 \end{aligned}$$

Course preference constraint:

$$x_{9,4} \geq 1$$

Prior course constraint:

$$x_{3,2} + x_{3,3} \geq 1$$

Total credit constraint:

$$F(x) = \sum_i \sum_j C_i x_{i,j} \geq 24$$

Objective function:

$$\text{Minimize } F(x) = \sum_i \sum_j C_i x_{i,j}$$

Solutions: The selections are presented in Table II, and for the remainder of the next stages of semester and module selection, the first course selection shown in Table II will be used.

B. Semester selection

With the first selection from Table II, the prioritized course list is as shown in Figure 4. Every semester is restrained to have a maximum of six credit hours. Semester 1 is treated as the next semester while all courses are unavailable in semesters 3 and 6 (to represent summer course unavailability). The previously taken course 3 is symbolically shown as having been taken in semester 0.

The tables that follow show the results of execution after each iteration.

Iteration 1:

Semesters							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	1						

Iteration 2:

Semesters							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	1	2					

Iteration 3:
All courses are unavailable – no change.

Iteration 4:

Semesters							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	1	2		4			
				5			

Iteration 5:

Semesters							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	1	2		4	9		
				5	6		

Iteration 6:
All courses are unavailable – no change.

Iteration 7:
Note that if course 2 happened to have been taken previously, then the time-to-degree for this student would have been shortened by two semesters, but instead there is a smaller workload in semester 2. Moreover, since the student was explicitly interested in course 9 it was scheduled in semester 5 instead of semester 7.

Semesters							
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
3	1	2		4	9		7
				5	6		

C. Module selection

Module selection is similar to course selection, it involves the selection of course modules based the student's interests and prior knowledge. Table III shows the module options for each course, and subsequent subsections show how the selection problem is modeled. Note that each module is worth one credit, each course requires three credit hours, and required modules are underlined.

TABLE III
COURSE MODULES.

Course 1	Needs three credit hours
Module Options	1,2,3,4
Course 2	Needs three credit hours
Module Options	5,6,7,8
Course 3	Needs three credit hours
Module Options	9,10,11,12
Course 4	Needs three credit hours
Module Options	13,14,15,16
Course 5	Needs three credit hours
Module Options	9,17,18
Course 6	Needs three credit hours
Module Options	13,19,20,21
Course 7	Needs three credit hours
Module Options	15,22,23,24
Course 8	Needs three credit hours
Module Options	17,25,26,27
Course 9	Needs three credit hours
Module Options	18,28,29,30

Decision variables:

$$\begin{aligned} X = \{x_{i,j} : & x_{1,1}, x_{2,1}, x_{3,1}, x_{4,1}, x_{5,2}, \\ & x_{6,2}, x_{7,2}, x_{8,2}, x_{9,3}, x_{10,3}, \\ & x_{11,3}, x_{12,3}, x_{13,4}, x_{14,4}, x_{15,4}, \\ & x_{16,4}, x_{9,5}, x_{17,5}, x_{18,5}, x_{13,6}, \\ & x_{19,6}, x_{20,6}, x_{21,6}, x_{15,7}, x_{22,7}, \\ & x_{23,7}, x_{24,7}, x_{18,9}, x_{28,9}, x_{29,9}, x_{30,9}\} \end{aligned}$$

where i = module, j = course.

Required modules constraints:

$$\begin{aligned} x_{1,1} + x_{2,1} + x_{5,2} + x_{6,2} + x_{10,3} &\geq 5 \\ x_{14,4} + x_{18,5} + x_{19,6} + x_{22,7} + x_{25,8} &\geq 5 \\ x_{9,3} + x_{9,5} &\geq 1 \\ x_{13,4} + x_{13,6} &\geq 1 \\ x_{15,4} + x_{15,7} &\geq 1 \\ x_{17,5} + x_{17,8} &\geq 1 \end{aligned}$$

Course credit constraints:

$$\begin{aligned} M_1x_{1,1} + M_1x_{2,1} + M_1x_{3,1} + M_1x_{4,1} &\geq C_1 = 3 \\ M_2x_{5,2} + M_2x_{6,2} + M_2x_{7,2} + M_2x_{8,2} &\geq C_2 = 3 \\ M_3x_{9,3} + M_3x_{10,3} + M_3x_{11,3} + M_3x_{12,3} &\geq C_3 = 3 \\ M_4x_{13,4} + M_4x_{14,4} + M_4x_{15,4} + M_4x_{16,4} &\geq C_4 = 3 \\ M_5x_{9,5} + M_5x_{17,5} + M_5x_{18,5} &\geq C_5 = 3 \\ M_6x_{13,6} + M_6x_{19,6} + M_6x_{20,6} + M_6x_{21,6} &\geq C_6 = 3 \\ M_7x_{15,7} + M_7x_{23,7} + M_7x_{24,7} + M_7x_{25,7} &\geq C_7 = 3 \\ M_9x_{18,9} + M_9x_{28,9} + M_9x_{29,9} + M_9x_{30,9} &\geq C_9 = 3 \end{aligned}$$

where M_i represents the credit value for module i.

TABLE IV
SOLUTION: MODULE SELECTIONS

Course	Modules Selected
1	1,2,3
2	5,6,7
3	10,11,12
4	13,14,15
5	9,17,18
6	19,20,21
7	22,23,24
8	28,29,30

Course reusability constraints:

$$x_{9,3} + x_{9,5} \leq 1$$
$$x_{13,4} + x_{13,6} \leq 1$$
$$x_{15,4} + x_{15,7} \leq 1$$
$$x_{18,5} + x_{18,9} \leq 1$$

Module preference constraints:

$$x_{29,9} + x_{30,9} \geq 2$$

Note that constraints for modules 18 and 28 do not need to be added because these modules are already required.

Module preference constraints:

$$x_{11,3} + x_{12,3} \geq 2$$

Note that constraints for modules 9 and 10 do not need to be added because these modules are already required.

Total credit constraint:

$$F(x) = \sum_i \sum_j M_i x_{i,j} \geq 24$$

Objective function:

$$\text{Minimize } F(x) = \sum_i \sum_j M_i x_{i,j}$$

Solution:

V. CONCLUSION

The primary contribution of this research was the creation of a methodology that facilitates student advising by identifying a course schedule conformant to curriculum requirements, with consideration of a student's background, interests, and desired time-to-degree.

One avenue for expanding this work lies within personalizing course schedules for the students who do not specify their interests. A CSA could consider previous students who were similar to the target student in order to find opportunities to personalize and reduce the number of decision variables. For example, elective courses in which similar students did especially well might be worth recommending to a target student. What constitutes similarity is also an open question though it requires a long time to measure. Different similarity metrics could be compared in the short term if applied to module selection instead. A final promising extension to this work involves solving the course selection problem as a multi-objective optimization problem. Multi-objective optimization problems seem to be a far more natural representation of the course selection problem since each AoE is really a competing objective. This research handles this problem by breaking the problem up into stages. However, exploring more sophisticated solutions in evolutionary algorithms seems promising as it would allow the simultaneously consideration of multiple objective functions, such as those for expected performance (if an appropriate prediction is used) and interests. Considering multiple objective functions immediately allows the researcher to effectively consider each AoE within the context of the others, rather than intelligently, but one-at-a-time as presented in this work. If the multi-objective optimization approach were adopted, the number of AoEs could be expanded in exciting ways to include accommodating deficiencies, preferred learning styles, and special needs. Moreover, constraint derivation for the semester selection stage could be replaced with an objective function and intelligent constraint derivation for the prerequisite, corequisite, and consequisite relationships.

ACKNOWLEDGMENT

The authors gratefully acknowledge financial support from the US National Science Foundation and Department of Education.

REFERENCES

[1] W. Murray, L. Blanc, and C. Rucks, "A decision support system for prescriptive academic advising," in *Proceedings of the 1995 ACM Symposium on Applied Computing*. ACM, 1995, pp. 22–26.

[2] W. Murray, L. LeBlanc, and C. Rucks, "A decision support system for academic advising," *Journal of End User Computing*, vol. 12, no. 3, pp. 38–49, Jul 2000.

[3] R. Hashemi and J. Blondin, "SASSY: A Petri net based student-driven advising support system," in *2010 Seventh International Conference on Information Technology: New Generations*. IEEE, 2010, pp. 150–155.

[4] S. M. Ross, G. R. Morrison, and D. L. Lowther, "Educational technology research past and present: Balancing rigor and relevance to impact school learning," *Contemporary Educational Technology*, vol. 1, no. 1, pp. 17–35, 2010.

[5] M. Erdt, A. Fernández, and C. Rensing, "Evaluating Recommender Systems for Technology Enhanced Learning: A Quantitative Survey," *IEEE Transactions on Learning Technologies*, vol. 8, no. 4, pp. 326–344, Oct. 2015.

[6] A. Bozkurt and A. Hilbelink, "Paradigm Shifts in Global Higher Education and e-learning: An ecological perspective," *eLearn*, vol. 2019, no. 5, p. 1, May 2019.

[7] A. Hurson and S. Sedigh, "PERCEPOLIS: Pervasive Cyberinfrastructure for Personalized Learning and Instructional Support," *Intelligent Information Management*, vol. 2, no. 10, pp. 583–593, Oct. 2010.

[8] T. Morrow, S. Sedigh Sarvestani, and A. Hurson, "Pervasive cyberinfrastructure for personalized education," *Handbook of Research on Applied Learning Theory and Design in Modern Education*, vol. 2, pp. 817–839, 2016.

[9] V. Swaminathan and R. Sivakumar, "A comparative study of recent ontology visualization tools with a case of diabetes," *International Journal of Research in Computer Science*, vol. 2, no. 3, p. 31, 2012.

[10] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72–79, 2001.

[11] M. Bright, A. R. Hurson, and S. Pakzad, "Automated resolution of semantic heterogeneity in multidatabases," *ACM Transations on Database Systems*, vol. 19, no. 2, pp. 212–253, 1994.

[12] R. Lougee-Heimer, "The common optimization interface for operations research," *IBM Journal of Research and Development*, vol. 47, no. 1, pp. 57–66, 2003.