ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 9, No. 4, October 2021, pp.632-642

A decision support system for curricula design

Sencer Yeralan[1,2], Özge Büyükdağlı[3]
[1]Computer Organization and Architecture, ADA University
[2] Agricultural and Biological Engineering, University of Florida
[3]Computer Sciences and Engineering, International University of Sarajevo

ABSTRACT

A curriculum is a set of related courses that constitutes the basis of a degree program. The required courses of a curriculum generally build student knowledge and skills particular to the field. In most cases, these are cumulative, meaning that as students go through their studies, they put their new knowledge on top of earlier ones, hence leading to the notion of prerequisite courses that must precede a given course. As accreditation practices gain widespread acceptance, and as uniformity among peer institutions is promoted to facilitate mobility, each course is assigned a set of learning outcomes. The learning outcomes of a prerequisite course are seen to encapsulate the skills necessary to take the downstream course. This study follows our efforts regarding the substantial revision of engineering courses throughout our college. As the task is quite involved, we developed a flexible linear programming based tool to help the decision making process by quickly evaluating alternative curricula. This study aims to provide an effective decision making tool to accommodate many "what if" scenarios which would provide options to the decision makers and help them detecting inconsistencies and oversights. This paper describes our approach and our experiences.

Keywords: Curricula design, Learning outcomes, Prerequisite courses, Decision support

Corresponding Author:
Sencer Yeralan,
Computer Organization and Architecture,
ADA University,
61 Ahmadbay Agha-Oglu Street, Baku, Azerbaijan, AZ1008
E-mail: syeralan@ada.edu.az

1. Introduction

Engineering curricula, as in many other fields, are built around a set of core courses. In this study, we considered engineering programs in International University of Sarajevo (IUS) for our analysis. We observe that the number of core courses range between 10 to 15 in each engineering curriculum. Each course in any curriculum has learning outcomes that defines the aim of that course. Some required courses have prerequisite courses that must be taken prior to that specific course. Examples are the dual-term courses (e.g. Operations Research I and Operations Research II), where two courses cover one large topic and divided into two consecutive terms, or strongly coupled courses (e.g. Statics and Strength of Materials), where the prior course provides the skills needed to study the following course. We further clarify and seek a higher resolution regarding the nature of prerequisite conditions. We scan the learning outcomes of the prerequisite courses to identify set of finer elements and consider these as prerequisite skills. This approach gives a finer understanding of the relations among the sequence of courses and allows more precision.

Our objective is to find feasible curricula that achieve a set of possible objectives. For example, we may consider evenly distributing the required courses among the terms. Similarly, we may want to force two courses to be given in the same term (e.g. Introduction to Engineering and Engineering Graphics) or to be separated in time as much as possible (e.g. Calculus and Systems Design). The advantage of flexible LP

model is to be able to introduce different objectives using the model body, via constraints. Even in the single objective LP models, flexibility in the design of the constraints provides the decision maker to apply additional objectives and different system characteristics.

Prerequisite-course-triggered precedent relations naturally lead to a network-like structure. The nature of the relations is linear, and thus quite appropriate for mixed integer programming (MIP) with binary decision variables. MIP not only gives a convenient media to model, present, and test various scenarios and work ”what if” cases, but it also enjoys the ready availability of capable and sophisticated software tools. We use the open-source GNU Linear Programming Kit (GLPK) package with a front end written in GNU Compiler Collection (GCC) for all of our design decision support tasks while revising the curricula.

2. Related literature

The assignment of courses to teaching periods while satisfying prerequisite relations and balancing term loads has been a common concern in academia. There are newer attempts to use formal models to address this concern. Introduced by the Balanced Academic Curriculum Problem (BACP) ([1]), these models typically use Integer Programming (IP) formulations, since most models have an underlying network topology stemming from prerequisite relations. A common objective function is to minimize the maximum student work load for each term. In these studies, the models come with simplifying assumptions, whereas curriculum design includes many external beyond the domain of such quantitative models.

Several variants of the BACP are found in the literature dealing with the performance and solution quality ([2], [3]). Interestingly, many studies focus on solution techniques, e.g. constraint programming (CP) aimed at the reduction of computational effort. Hnich et al. [4] propose different ways to model this problem. They present a model where the problem domain may be pruned and the run-times be reduced. Di Gaspero et al. [5] extend BACP by adding professor preferences and call this problem the Generalized Balanced Academic Curriculum Problem (GBACP). Unal et al. [6] propose the so-called Relevance Based Curriculum Balancing (RBCB) problem where they assign relevant courses to the closest possible periods while meeting all of the constraints of BACP. This study is one of the few studies in the literature that considers the relationship between courses other than pre-defined prerequisite relations. They define relevance scores as the level of interdependency between courses. A 0–9 rating scale is used where score ‘9’ corresponds to a strong relationship and the scores get closer to ‘0’ if the relation is weak. They formulate this problem as a bi-objective Mixed Integer Linear Programming (MILP) model where the objective functions are to minimize the distance between the relevant courses and the deviation from the average workload per semester.

Another branch of the literature on curriculum design considers the curriculum and course relations as a network. Recently, this modeling perspective received more attention by researchers and administrators while designing and analyzing course plans and workloads of the students. Graph theory provides a holistic and quantitative perspective for curricula designers. Generally, the network representation of curricula is constructed by considering the courses as nodes, and the prerequisite relations as directed arcs. The study by Lightfoot [7] is among the earlier studies that use acyclic directed graph representations of curricula. They investigate some graph metrics such as in-degree, out-degree, measures of centralities and clustering and their relations to curriculum design. Aldrich [8] analyze the Benedictine University course catalog and its underlying network structure. He models the system as a directed acyclic graph to study the curriculum structure of the university. Slim et al. [9] introduce a framework to detect the courses with a high impact on students’ progress, and also to quantify the cruciality of these courses, using network analysis and graph theoretic concepts. They used the “cruciality” of courses in their formal model, which differs from the RBCB formulation.

Knorn et al. [10] present a different network structure than earlier studies, called Directed Courses-Concepts Graph (DCCG). They created two separate node sets: courses and concepts. Concepts are generated using the learning outcomes of the courses within the program. They then defined the links between concept nodes and course nodes to be either requirements or learning outcomes of the course, depending on the direction of the edge. This approach can be useful to detect the mismatches or redundancies in existing curricula, and can be

used while determining and assigning prerequisite relations. It is one of the few studies in the literature that models the system including the contents of the courses and relations between them as a learning flow.
There are also some studies using predictive models and curriculum visualization for curriculum generation. Akbas et al. [11] propose an adaptive curriculum generation and planning system, where the model is trained first by data from former students. The trained model is then used to create quantitative recommendations for individual current students, considering their status. Siirtola et al. [12] develop an effective tool to visualize the curriculum which aims to analyze the curriculum contents and try to detect overlaps with other programs.

3. Approach

Our approach is to provide a flexible tool quickly to test out various scenarios. Our model is thus more of a "*real-time calculator*" used for decision support, rather than a formal model to "*deliver a solution*". Our experience is that with commonly available software tools and modest laptop computers, these solutions are obtained in just a few seconds. Given the small problem sizes (only tens of courses), we see little need to reduce computational effort. It should be noted that we did try to limit all our decision variables to be binary variables. This assists modern optimization software in finding solutions in a very short time.
We make use of open source software, primarily the GLPK. We wrote a simple front-end pre-processor in GCC to quickly parse course information and prepare a data file for GLPK. The preprocessor spawns GLPK and the results are immediately observable. We view the GLPK code as part of the flexible approach, where we may freely insert additional constrains, or modify the objective function. As such, we have developed, and used in house, a practical and expedient decision support tool. As a decision support tool, the model does not claim to capture curricular intricacies. Such externalities are to be discussed by the faculty councils in the spirit of participatory and collective management of academic processes. The tool, however, is kept on hand and during deliberations to rapidly test out various "what-if" scenarios. A few examples of these modifications are given in Section 6.

3.1. Learning elements

Before we develop our mathematical model, we will first present and motivate the ingredients of our approach. The so-called learning outcomes (LO) describe the expected skills and competences the student acquires after successfully finishing a given course. The LOs are typically broad descriptions. We motivate our insertion with an example. Many engineering courses require the student to be skillful in calculus and numerical analysis. Let us consider such an engineering course: Strength of Materials (SoM). The prerequisite for SoM at IUS is Statics, and the prerequisite to Statics is Calculus 1. The LOs of Calculus 1 are given below:

1. Recognize and graph basic polynomial, rational and trigonometric functions.
2. Compute basic limits and have an understanding of the formal definition.
3. Use all the rules for computing derivatives and be familiar with the definition of derivatives and the tangent line.
4. Use derivatives to find maxima/minima of a function.
5. Use derivatives to determine the monotinicity or concavity, and graph functions.
6. Find basic anti-derivatives and compute definite integrals.

Similarly, the LOs of Statics are:

1. Construct free-body diagrams and calculate the reactions necessary to ensure static equilibria.
2. Analyze distributed loads.
3. Analyze internal forces and moments in membranes.
4. Conduct force analysis on structures.
5. Calculate centroids and moments of inertia.
6. Solve static equilibrium problems involving friction.

At a general level, the LOs of the prerequisite courses cover the skills needed to delve into SoM. However, the LOs are rather non-specific. For example, SoM often makes use of polar coordinates (e.g. in dealing with

shafts and cylindrical structures). Similarly, stress and strain considerations require relatively simple trigonometric functions (usually limited to only sine and cosine functions).

The observation we want to highlight is that prerequisite courses may be conducted in such a way that, say, polar coordinates are not at all covered, and the more involved trigonometric and hyperbolic functions are emphasized by an ambitious mathematics professor at the expense of simpler sine and cosine functions.[1]

We seek a finer granularity and more specificity of LOs by defining Learning Elements (LE). One may consider LEs as the contents or individual components of a given LO. Moreover, we consider LEs not only as outcomes, but as inputs, that is, as more detailed components of the prerequisite courses which are deemed necessary for enrolling in a downstream course.

Next, we establish the inter-dependency of courses through input and output LEs. This inter-dependency preserves the prerequisite relations, but provides the enhanced granularity we seek.

4. The model

4.1. Model objectives

We build a computational mathematical model to serve us in the design of the curriculum of a single program, or concurrently, for a set of curricula of related engineering programs. Treating multiple engineering programs together allows the extraction of further efficiencies by better coordinating and synchronizing courses common to different programs. We would like to use the model as a decision support tool, where several scenarios are tested and examined. We realize that curricula design demands more than the mechanical matching of course inputs and outputs. Curricula must also consider social and strategic priorities of the institution. This relegates the model to a convenient computational support tool, rather than an ultimate mechanism to produce the designs. That is, our model is a design support tool rather than a design automation tool. The choice of using MIP follows from its flexibility. It is straightforward to add new constraints, or to change objective functions in MIP. Since the size of the problems we anticipate are small (only tens of courses and hundreds of learning outcomes), the computational effort demanded from modern MIP software is negligible. Provide sufficient detail to allow the work to be reproduced. Methods already published should be indicated by a reference: only relevant modifications should be described.

4.2. The model

Each course is associated with a set of input LEs and a set of output LEs. All input LEs are deemed necessary for the eligibility of the subsequent course. The input LEs may be acquired from several other courses, not just a given prerequisite. We seek to create curricula by assigning courses to terms in a way that satisfies the input-output LE relations. Experience shows that several such assignments are usually possible. To further assist in decision-making, we consider several objectives. These include the uniform distribution of courses along the entire curricula, placing two given closely coupled courses one after another, or separating a set of given courses as far apart as possible. The generation of a rich set of alternatives allows the decision maker to entertain secondary, and rather qualitative concerns.

Moreover, the formulation may be applied to many engineering courses collectively. Although different engineering programs will have different subsets of required courses, there are nonetheless several common courses. Our experience shows that the collective consideration of many engineering courses has been helpful in coordinating between the various programs and reducing the need to teach common courses every term to accommodate the characteristics of the otherwise independently developed program curricula.

4.3. The mathematical model

We use *Mixed Integer Programming* (MIP) software to implement the model. There may be more efficient ways to find possible course assignments. However, the ready availability of software makes the use of MIP a practical and expedient choice. We use the open-source GLPK and developed a preprocessor to generate the input to GLPK from a higher level of abstraction. The preprocessor is written in C and compiled with the

[1] This, to our dismay, has happened recently in our Strength of Materials course.

GCC[2]. The preprocessor not only formats the input file, but also verifies that all LEs are used either as an input or an output, or both. Also, each input LE must be specified as an output LE of some other course. We run our experiments in a Linux environment on a rather unassuming laptop computer.

Table 1. Notation

Sets		
T	Terms	$t \in \{0,1,\dots,9\}$
L	Learning elements	$l \in \{1,2,\dots,E\}$
C	Courses	$c \in \{1,2,\dots,N\}$
Parameters		
N	Number of courses	
M	Maximum number of courses per term	
E	Number of learning elements (LE)	
$Q_{c,l}$	Binary parameter, 1 if LE l is an input of course c	$\forall c, \forall l$
$R_{c,l}$	Binary parameter, 1 if LE l is an output of course c	$\forall c, \forall l$
$BigM$	A sufficiently large number (e.g. 100)	
Binary Variables		
$A_{l,t}$	Learning element l is available at the beginning of the term t	$\forall l, \forall t$
Binary Decision Variables		
$X_{c,t}$	Course c is scheduled for term t	$\forall c, \forall t$

In the formulation, all variables are binary variables. This saves computational effort. All variables and parameters are denoted by single letters, except for the traditional "*big M*". Table 1 gives the notation.

The curriculum spans 8 terms (semesters, trimesters, etc.). However, we start the terms from 0, where term 0 is the start of the curriculum. We also use term 9 to indicate the conditions at graduation. Essentially, terms 0 and 9 are "virtual" terms that are used for boundary conditions. Similarly, we start the LEs from 0, which corresponds to the requirement that the student enrolls in the program. Any course that does not need any prerequisite LEs is designated with the input LE=0.

The size of the problems determined by the number of terms, the number of courses, and the number of learning elements is expected to remain quite small. The number of terms could be left as a parameter, but it is taken here as a constant, 8, following our standard four-year, two-semesters-per-year programs. The number of courses is around 10 to 15, while we expect 100 to 200 learning elements. Given the high performance of readily available software (GLPK can solve problems with tens of thousands of variables), not surprisingly, typical computation times are only a few seconds. We now present the formal model.

$$minimize\ Z = \sum_c \sum_t t \cdot X_{c,t} \tag{1}$$

$subject\ to$:

$$A_{l,0} = 0 \qquad l > 0 \tag{2}$$

$$A_{0,t} = 1 \qquad \forall t \tag{3}$$

$$X_{c,0} = 0 \qquad \forall c \tag{4}$$

$$X_{c,9} = 0 \qquad \forall c \tag{5}$$

$$A_{l,t} \le \sum_c \sum_{0<k<t} \left(X_{c,k} \cdot R_{c,l}\right) \qquad l > 0, t > 0 \tag{6}$$

$$BigM \cdot A_{l,t} \ge \sum_c \sum_{0<k<t} \left(X_{c,k} \cdot R_{c,l}\right) \qquad l > 0, t > 0 \tag{7}$$

$$X_{c,t} \le A_{l,t} + BigM \cdot \left(1 - Q_{c,l}\right) \qquad \forall c, l > 0, 0 < t < 9 \tag{8}$$

$$A_{l,9} \ge 1 \qquad l > 0 \tag{9}$$

[2] The source code of the preprocessor and the GLPK model are available from the authors upon request.

$$\sum_{t,} X_{c,t} \leq 1 \qquad \forall c \qquad (10)$$

$$\sum_{c} X_{c,t} \leq M \qquad \forall t \qquad (11)$$

$$A_{l,t} \in \{0,1\} \qquad \forall l, \forall t \qquad (12)$$

$$X_{c,t} \in \{0,1\} \qquad \forall c, \forall t \qquad (13)$$

The function (1) given above is but one possible such objective. It aims to finish all required courses as soon as possible. Other objective functions are discussed in Section 6.

The equations (2) to (5) set the boundary conditions, at the start and the end of the curriculum. LE 0 is satisfied and is available at the beginning (term 0). Moreover, no course is scheduled at term 0 or term 9. Equations (6) and (7) set the values of $A_{l,t}$. Equation (6) sets the upper limit of $A_{l,t}$ to a positive number (≥ 1) if l is an output LE of one of the courses scheduled up to and including term $t-1$. Equation (7) forces the binary variable $A_{l,t}$ to 1 if l is an output LE of one of the courses scheduled up to and including term $t-1$. Equation (8) guaranteed that a course is assigned to a term only if all its learning elements are available at the beginning of the term. Note that the inequality is automatically satisfied if the learning element l is not an input to the course. Equation (9) requires that all LEs are satisfied by the curriculum, equation (10) prevents a course from appearing more than once (in different terms) in the curriculum, and equation (11) sets the upper limit to the number of courses to be assigned to a single term. Finally, variable domains are given in (12) and (13).

The formulation provides the flexibility to conduct "what if" scenarios. For example, rather than requiring all LEs to be satisfied by the end of the curriculum, we may relax our requirements and see how the curriculum changes if we require that at least 90% of the LEs are required to be satisfied. Then equation (9) would be replaced by equation (14):

$$\sum_{l} A_{l,9} \geq 0.90 \cdot E \qquad (14)$$

5. Design rule check

The MIP formulation provides a natural means for all types of design rule checks. If there are courses with input LEs that are not provided as output LEs by any other course, for example, our curriculum would not be viable. In this case, the MIP solver would report that the problem is infeasible. Similarly, if there are cyclical prerequisites, the solver will report infeasibility. Thus, rather than listing and checking for separate forms of inconsistencies, as it has been reported in the literature (see Knorn et al. [10]), the model provides a means to check any kind of infeasibility in one fell swoop. This is an important contribution of our study, since not all cases with LE cycles are infeasible, and not all courses need be linked with LE input-output relations.

Beyond infeasibility, the model also provides means to highlight shortcomings. In this sense, rather than a design support tool, the model also functions as an evaluation support tool, bringing potential oversights into focus. Such a case is discussed in Section 7.2 below.

6. Alternative objectives and constraints

The MIP formulation gives us great latitude to implement various objectives and constraints. Only a few examples are given below.

6.1. Early completion of required courses

We may wish to set the curricula so that the required courses as completed as soon as possible. This will leave the latter years of the study program to pursue further specializations (or tracks). There are many ways to formulate this, each with a slightly different slant.

The original objective function (1) asks to complete the curricula as early as possible. An alternative may be implemented as follows.

$$minimize\ Z \tag{15}$$

$subject\ to$:

$$Z \geq t \cdot X_{c,t} \qquad \forall c, \forall t \tag{16}$$

Here, the objective function value Z is the term which completes the curriculum. In our case, we would like Z to be 6 or 7, leaving one or two terms for our specialization tracks. As stated, there are many other possible ways to implement similar objectives.

6.2. Restricting courses to terms

A common modification we used comes from the desire to restrict a given course to a certain period. Such a hard requirement is best implemented with additional constraints. Let course k be limited to the terms from u to v. The following additional constraints specify this requirement.

$$\sum_{u \leq t \leq v} X_{k,t} \geq 1 \tag{17}$$

Similarly, if we want course k not to be placed within the terms u to v, we have,

$$\sum_{u \leq t \leq v} X_{k,t} \leq 0 \tag{18}$$

6.3. Coordinating two courses

Another common modification regards forcing two courses, either to be scheduled in the same term, or in different terms. If the two courses k and m are to be scheduled in the same term, we simply may include the following constraints.

$$X_{k,t} \leq X_{m,t} \qquad \forall t \tag{19}$$

$$X_{k,t} \geq X_{m,t} \qquad \forall t \tag{20}$$

If, on the other hand, we would like these courses not to be offered in the same term, we simply add the following constraint.

$$X_{k,t} + X_{m,t} \leq 1 \qquad \forall t \tag{21}$$

This constraint can easily be extended to cases where, say, at most two of the three courses k, l, and m may be scheduled in the same term.

$$X_{k,t} + X_{l,t} + X_{m,t} \leq 2 \qquad \forall t \tag{22}$$

7. Implementation

We used the model in our efforts to review and re-design engineering curricula. We gain experience in both designing individual engineering programs, and also in examining several engineering programs concurrently. The multi-program case helps in coordinating between individual programs. Most importantly, we wanted the common courses to be offered once in the same term (either Fall or Spring, but not both).

7.1. Designing a single program

Table 2 lists the 23 core courses of the Computer Sciences and Engineering (CSE) program. The elective courses and required humanities courses are excluded from the list. There are 106 LEs, with the addition of LE 0, which corresponds to the “no prerequisite” condition.

Table 2. CSE Core Courses and LEs

Course	Input LEs	Output LEs
CS103	0	1,2,3,4,5
CS105	1,2,3,4,5	6,7,8,9
CS302	3,4,5,7,8,9,79,81,83	10,11,12,13,14
CS303	97	15,16,17,18,19
CS304	7,16,17,18	20,21,22,23
CS305	1,3,4,5,6,7,8,9,10	24,25,26,27
CS306	1,5,8,9,10	28,29,30,31,32
CS307	1,2,3,4,5,6,7,8,9,10,14,20,21,22	33,34,35,36,37
CS308	1,2,6,7	38,39,40,41,42
CS313	6,7,9,10,11,13,14,20,21,26,27,79,80,81	43,44,45,46
CS370	42,43	47,48,49,50,51
CS412	1,2,3,4,5,6,7,8,9,10,11,14,22,26	52,53,54
EE325	1,2,3,4,5	103,104,105,106
ENS203	65	97,98,99,100,101,102
ENS490	52	53
MATH101	0	65,66,67,68,69,70
MATH102	65,66,67,68,69,70	71,72,73,74,75
MATH201	65	76,77,78
MATH202	67,68,69,70,71,73,74,76	94,95,96
MATH203	65,66	84,85,86,87,88
MATH204	65	79,80,81,82,83
MATH205	65,66,67,68,69,74,76	89,90,91,92,93
SE308	16,17,20,33,37	60,61,62,63,64

The data was fed into the GLPK solver, setting the maximum courses per semester to 4 and 5. The default objective function was used, striving to complete the core curricula as early as possible (Table3).

Table 3. CSE core curricula

Term	Maximum 4 courses per term				
1	CS103	MATH101			
2	CS105	MATH102	MATH201	ENS203	
3	CS302	CS303	CS308	MATH204	
4	CS304	CS305	MATH202	EE325	
5	CS306	CS307	CS313	CS412	
6	CS370	ENS490	SE308	MATH203	
7	MATH205				
Term	Maximum 5 courses per term				
1	CS103	CS303	MATH101		
2	CS105	ENS203	MATH102	MATH201	MATH204
3	CS302	CS304	EE325	MATH202	MATH203
4	CS305	CS306	CS307	CS308	MATH205
5	CS313	CS412	SE308		
6	CS370	ENS490			

Both of these curricula were discussed and were seen to have merit.

7.2. Design evaluation

The pragmatic use of the model as a real-time support tool allows the verification of curricula design. We offer as an example the detection of missing LE relations among courses. Courses CS303 and ENS203 deal with digital design and electrical circuits, respectively. Earlier, CS303 did not have an input LE. When the

initial curriculum was generated and presented to the faculty members, objections were raised. The oversight was then noticed as not having specified the output LEs of electrical circuits as input LEs to digital design. Table 4 shows the curriculum generated using the original set of LEs and the corrected one, where an output LE 97 of ENS203 is specified as an input LE of CS303, as shown in Table 2.

The model may be used to test out LE relations and uncovering missing or superfluous relations. Generating alternative curricula with different objective functions would help identifying such oversights.

Table 4. Identifying missing LEs (see text).

Term	Missing LE '97'			
1	CS103	CS303	MATH101	
2	CS105	EE325	MATH204	MATH203
3	CS302	CS304	MATH102	MATH201
4	CS305	CS307	CS308	MATH202
5	CS306	CS313	CS412	SE308
6	CS370	ENS203	ENS490	MATH205
Term	Corrected LEs			
1	CS103	MATH101		
2	CS105	MATH102	MATH201	ENS203
3	CS302	CS303	CS308	MATH204
4	CS304	CS305	MATH202	EE325
5	CS306	CS307	CS313	CS412
6	CS370	ENS490	SE308	MATH203
7	MATH205			

As an additional observation, it is interesting to see how one LE can create a domino effect and result in a number of changes in the curriculum.

7.3. Concurrently designing multiple programs

The model may be used to synchronize multiple curricula. Here we consider Computer Sciences and Engineering (CSE) and Software Engineering (SE), two programs that share a considerable number of courses. The following table gives the required core courses for each program.

Table 5. CSE and SE courses

Common Courses	CSE only	SE only
CS103	CS303	CS310
CS105	CS313	CS420
CS302	CS370	MATH209
CS304	EE325	SE211
CS305	ENS203	SE302
CS306	MATH102	SE322
CS307	MATH202	SE406
CS308	MATH205	SE407
CS412		
ENS490		
MATH101		
MATH201		
MATH203		
MATH204		
SE308		

There are 15 courses common to both programs. CSE and SE both have an additional 8 required courses that differ. It is desired that CSE and SE students take the common courses jointly. That is, the common courses

should appear only once per academic year, and not be repeated for CSE and SE separately. Since the total number of courses is 31, we attempt to create a curriculum with all courses combined. Table 6 shows the resultant curriculum using the default objective function.

Table 6. Combined CSE / SE curricula (see text).

Term	Combined CSE / SE Curricula				
1	CS103	MATH101			
2	CS105	EE325	ENS203	MATH204	SE211
3	CS302	CS303	MATH201	MATH203	SE302
4	CS304	CS305	CS308	SE322	SE407
5	CS306	CS307	CS412	MATH102	SE406
6	CS313	CS370	ENS490	MATH209	SE308
7	CS310	CS420	MATH202	MATH205	

From the combined CSE/SE curricula, each program may pick the courses required and leave out the courses. The resultant curricula have common courses shared and assigned to the same terms, thereby achieving the desired efficiency.

Table 7. CSE and SE curricula

Term	CSE Curriculum				
1	CS103	MATH101			
2	CS105	EE325	ENS203	MATH204	
3	CS302	CS303	MATH201	MATH203	
4	CS304	CS305	CS308		
5	CS306	CS307	CS412	MATH102	
6	CS313	CS370	ENS490	SE308	
7	MATH202	MATH205			
Term	SE Curriculum				
1	CS103	MATH101			
2	CS105	ENS203	MATH204	SE211	
3	CS302	MATH201	MATH203	SE302	
4	CS304	CS305	CS308	SE322	SE407
5	CS306	CS307	CS412	SE406	
6	ENS490	MATH209	SE308		
7	CS310	CS420			

8. Conclusions and contribution

Modern optimization software has evolved to a point where its use as a general-purpose tool in design and decision support is quite expedient. Their routine use is further enhanced by superb open-source software. Coupled with the computation power of even unassumingly ordinary laptop computers, using MIP software for decision and design support becomes practical.

We develop a base MIP model and use it in curriculum revision and design. The flexibility of MIP formulations allows the base model to be modified or tweaked to accommodate many "what if" scenarios. The objective is not to completely automate curriculum design, but to provide options to the decision makers. The model also helps in detecting inconsistencies and oversights, as demonstrate in Section 5. In our case, following our college practice of transparent and collective management, we make the scenarios available to all interested faculty members and use the model as a convenient tool during brainstorming and strategy meetings.

The contributions of this work are twofold. First, it introduces the concept of Learning Elements (LE) that are used to identify both the inputs and outputs of courses. These LEs, allow finer granularity in describing prerequisite relations among courses. They also allow the concurrent consideration of curricula from multiple

programs towards further synchronization and the larger-scale optimization across programs. Our work illustrates the benefits of using formal quantitative models in curricula design. As a second contribution, our work serves as an example of a state-of-the-art MIP formulation that may be used as a foundation for similar studies.

Acknowledgments

We acknowledge the contribution of our academic staff who throughout the revision of curricula advanced many comments and questions that greatly helped the formulation of our model and improved its relevance in a realistic environment.

References

[1] C. Castro and S. Manzano, "Variable and value ordering when solving balanced academic curriculum problems", *arXiv preprint cs/0110007*, 2001.

[2] J.-N. Monette, P. Schaus, S. Zampelli, Y. Deville and P. Dupont, "A CP approach to the balanced academic curriculum problem", *Seventh International Workshop on Symmetry and Constraint Satisfaction Problems*, vol. 7, 2007.

[3] M. Chiarandini, L. Di Gaspero, S. Gualandi and A. Schaerf, "The balanced academic curriculum problem revisited", *Journal of Heuristics*, vol. 18, no. 1, pp. 119–148, 2012.

[4] B. Hnich, Z. Kiziltan and T. Walsh, "Modelling a balanced academic curriculum problem", *Proceedings of CP-AI-OR-2002*, pp. 121–131, 2002.

[5] L. Di Gaspero and A. Schaerf, "Hybrid local search techniques for the generalized balanced academic curriculum problem", *International Workshop on Hybrid Metaheuristics*, Springer, pp. 146–157, 2008.

[6] Y. Z. Unal and O . Uysal, "A new mixed integer programming model for curriculum balancing: Application to a turkish university", *European Journal of Operational Research*, vol. 238, no.1, pp. 339–347, 2014.

[7] J. M. Lightfoot, "A graph-theoretic approach to improved curriculum structure and assessment placement", *Communications of the IIMA*, vol.10, no.2, 2010.

[8] P. R. Aldrich, "The curriculum prerequisite network: Modeling the curriculum as a complex system", *Biochemistry and Molecular Biology Education*, vol. 43, no.3, pp. 168–180, 2015.

[9] A. Slim, G. L. Heileman, E. Lopez, H. Al Yusuf and C. T. Abdallah, "Crucial based curriculum balancing: A new model for curriculum balancing", *10th International Conference on Computer Science & Education (ICCSE), IEEE*, pp. 243–248, 2015.

[10] S. Knorn, D. Varagnolo, K. Staffas, T. Wrigstad and E. Fjallstrom, "Quantitative analysis of curricula coherence using directed graphs", *IFAC PapersOnLine*, vol. 52, no. 9, pp. 318–323, 2019.

[11] M. I. Akbas, P. Basavaraj and M. Georgiopoulos, "Curriculum GPS: an adaptive curriculum generation and planning system", *Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*, pp. 1–11, 2015.

[12] H. Siirtola, K.-J. Raiha and V. Surakka, "Interactive curriculum visualization", *17th International Conference on Information Visualisation, IEEE*, pp. 108–117, 2013