The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #41711

Minimizing Curricular Complexity through Backwards Design

Prof. Gregory L. Heileman, The University of Arizona

Gregory (Greg) L. Heileman currently serves as the Vice Provost for Undergraduate Education and Professor of Electrical and Computer Engineering at the University of Arizona, where he is responsible for facilitating collaboration across campus to strategically enhance quality and institutional capacity related to undergraduate programs and academic administration. He has served in various administrative capacities in higher education since 2004.

Professor Heileman currently serves on the Executive Committee of AZTransfer, an organization that works across the system of higher education in the State of Arizona to ensure students have access to efficient, seamless, and simple ways to transfer from a community college to a university in Arizona. He serves on the board of the Association for Undergraduate Education at Research Universities, a consortium that brings together research university leaders with expertise in the theory and practice of undergraduate education and student success. In addition, he is a fellow at the John N. Gardner Institute for Excellence in Undergraduate Education.

Professor Heileman's work on analytics related to student success has led to the development of a theory of curricular analytics that is now being used broadly across higher education in order to inform improvement efforts related to curricular efficiency, curricular equity, and student progression.

Dr. Yiming Zhang, The University of Arizona

Yiming Zhang completed his doctoral degree in Electrical and Computer Engineering from the University of Arizona in 2023. His research focuses on machine learning, data analytics, and optimization in the application of higher education.

Minimizing Curricular Complexity through Backwards Design

Gregory L. Heileman and Yiming Zhang

{heileman, yimingzhang1}@arizona.edu
Department of Electrical & Computer Engineering
University of Arizona

Abstract

In this paper, we first describe the Optimal Learning Outcomes Assignment (OLOA) problem, which involves assigning learning outcomes to courses during the backwards curriculum design process in ways that minimize the complexity of the resulting curriculum. An approximation algorithm for the OLOA problem is then described that yields novel solutions to important engineering curricular design challenges. Reducing curricular complexity, while maintaining effective learning outcomes attainment, increases the likelihood students will complete a curriculum and earn a degree. The rationale for the approach taken here follows from the fact that by rearranging the learning outcomes among the courses in a curriculum, the overall structure of a curriculum can be changed. Thus, the OLOA problem provides a criterion for finding curricular structures that enhance student success. The OLOA problem is shown to be strongly $\mathcal{NP}$-complete; however, an integer quadratic programming approximation algorithm is described that effectively produces practical, efficient, and novel solutions for attaining the most important leaning outcomes in an undergraduate engineering curriculum.

Introduction

The use of learning outcomes in higher education is now a ubiquitous part of continuous quality improvement efforts. Learning outcomes are statements of what a learner is expected to know, understand and be able to demonstrate at the end of some learning experience. For instance, ABET stipulates a minimal set of student learning outcomes that describe what learners should know and be able to by the time they graduate from an ABET-accredited engineering program.[1] It is also now common practice to articulate course-level learning outcomes for each of the courses offered by a college or university; these indicate what a learner is expected to know and be able to do after successfully completing a course. A common approach used by curriculum designers, known as *backwards design*, involves designing a curriculum from the bottom up by starting from the program learning outcomes, and then creating course-level objectives that would allow the program-level learning outcomes to be attained, i.e., a curriculum map. Finally, specific learning modules are created to allow the course objectives to be achieved. In this paper, backwards

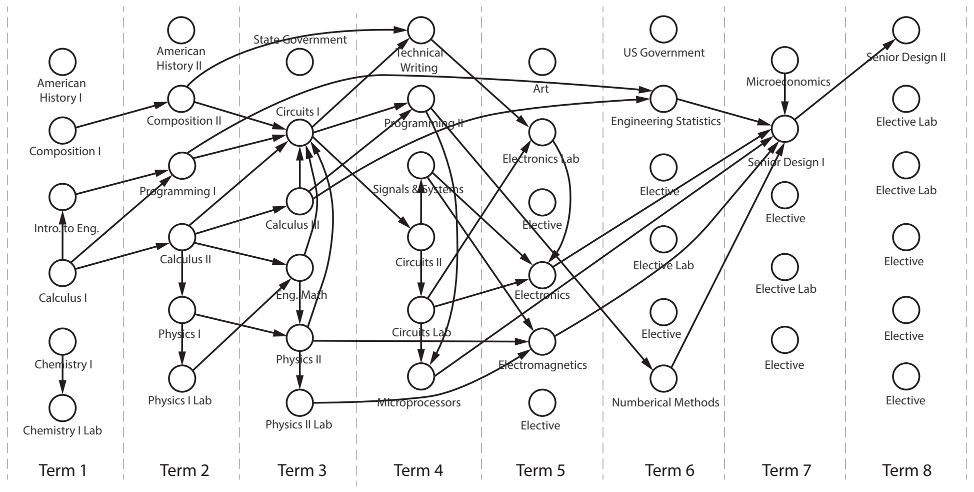

Figure 1: A dag representation of an electrical engineering curriculum

design techniques are considered that involve optimizing the arrangement of course-level learning outcomes within engineering curricula in ways that lead to improved student success outcomes.[2,3] Specifically, engineering programs offer some of the most complex curricula in higher education, and this complexity negatively impacts various student success metrics, including time-to-degree and graduation rates. Thus, the ability to reduce curricular complexity through backwards design is a significant consideration for engineering programs, and that should inform the work of curriculum committees.

1 Background

A curricular analytics framework was developed in order to quantify the various components that contribute to the complexity of a curriculum.[4] Broadly, this framework partitions these components into two categories, those that are based on how a curriculum is structured, and those that are based on how instruction is offered and supported within the curriculum. The former category is referred to as the *structural complexity*, and the latter as the *instructional complexity* of the curriculum. For computing structural complexity, a useful representation of a curriculum is as a directed acyclic graph (dag), where the vertices of the graph represent the courses in the curriculum, and the directed correspond to the prerequisites relationships among the courses in the curriculum. An example degree plan for an electrical engineering program, created using this dag representation is shown in FIgure 1. The structural complexity measure we will use is based upon the properties of this dag, and it has been shown that it directly relates to a student ability to complete the curriculum.[4] Specifically, two important graph properties determine the structural complexity of a course in the curriculum. The first is the longest path that goes through a course, which defines the *delay factor* of that course in the curriculum. The second is the number of other courses that are reachable from a given course, which defines the *blocking factor* the course. The, the sum of delay blocking factors, across all courses in a curriculum, defines the structural complexity of that curriculum.

It is important to recognize the critical role course prerequisites play in determining the structural

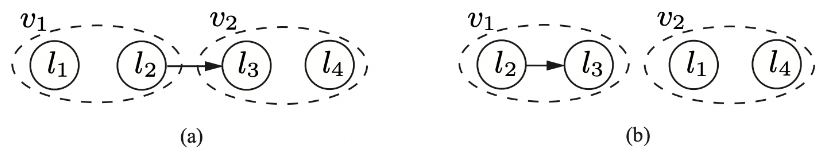

Figure 2: Two different curricula organized around the attainment of four course-level learning outcomes, l_1, l_2, l_3, and l_4, where attainment of l_3 requires a learner to first attain l_2. (a) If l_1 and l_2 are placed in one course, and l_3 and l_4 in the other, a prerequisite constraint is created between courses v_1 and v_2, and the structural complexity of the pattern is 5. (b) If l_2 and l_3 are placed in one course, and l_1 and l_4 in the other, then all prior required learning is contained within course v_1, no course-level prerequisites are required, and the structural complexity of the pattern is 2.

complexity of a curriculum. Furthermore, if we consider the same engineering program at different institutions, we will find there is often significant variation among the structural complexities of the curricula offered by these programs. As mentioned previously, those with lower structural complexity will have higher completion rates, assuming all other factors, e.g., instructional complexity and student preparation, are held constant. Somewhat counterintuitively, it has also been shown that in addition to improving graduation rates, in some disciplines, engineering programs with lower structural complexity are also judged to have higher quality.[5]

This leads us to consider the rationale for prerequisites in a curriculum. Presumably, a prerequisite from course A to course B in a curriculum is stipulated if there is some learning outcome in course A that must be attained prior to attempting course B. That is, prior knowledge obtained in course A is necessary for a student to succeed in course B. In other words, it is actually the relationships between the underlying course-level learning outcomes that creates the necessity for a prerequisite between two courses. To better understand this point; that is, how a learning outcomes dag impacts a curriculum dag, consider a simple case involving only four learning outcomes, denoted l_1, l_2, l_3, and l_4, where attainment of l_3 requires a learner to first attain l_2. In Figure 2, we show two possible arrangements of these learning outcomes into courses that lead to two different curricula. The arrangement in Figure 2 (a) leads to the creation of a course-level prerequisite in the curriculum, while in the case shown in Figure 2 (b), no course-level prerequisites are required. Thus, the curriculum in Figure 2 (b) has lower structural complexity than the one in Figure 2 (a), while attaining the same learning outcomes. An important point to note is that as the size of a learning outcome dag grows larger, say in a linear fashion, the number of curricula that can be created from the learning outcome dag grows exponentially, quickly overwhelming our ability to manually construct and consider alternative curricula for a single learning outcome dag. However, this is precisely what we need to do as a part of backward curriculum design. A key concept that will be utilized in this work is the fact that course-level prerequisites are dictated by the prerequisites in the underlying learning outcomes graph. Thus, the judicious arrangement of learning outcomes within courses may allow one to arrange the prerequisites in a curriculum in a way that reduces curricular complexity. Below, methods for automating this process in an optimal fashion are considered.

In order to better understand how the backwards design process can be automated, it is important to consider the solution space in more detail. In Figure 3 (a) we show an example learning outcomes graph. In order to construct a curriculum, we must assign learning outcomes to courses in the

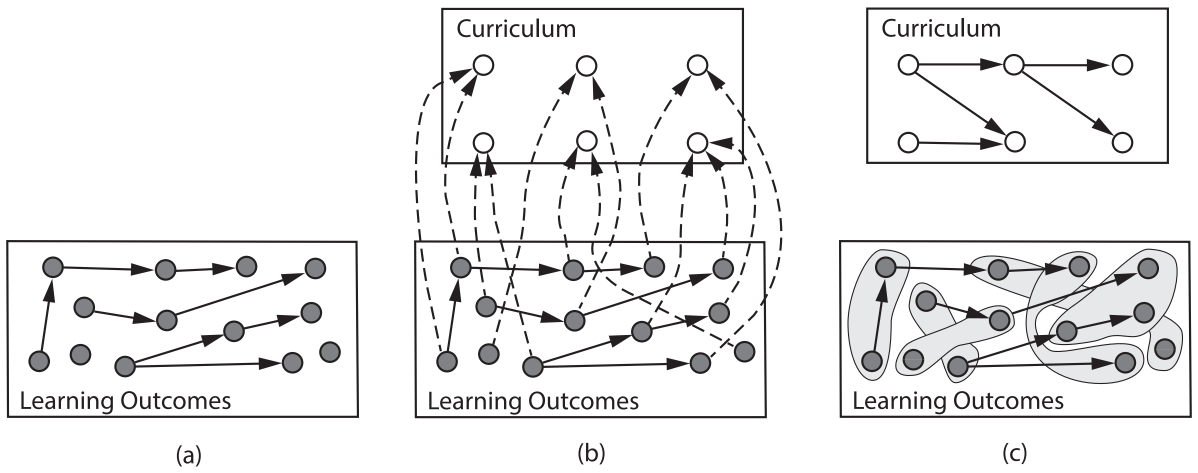

Figure 3: **(a)** A learning outcomes directed acyclic graph (dag), followed by **(b)** a mapping of these learning outcomes to courses, and finally **(c)** the curriculum dag imposed by the mapping.

curriculum, as shown in Figure 3 (b). Notice that the dashed edges in this figure actually constitute another dag, one that maps learning outcomes to courses. In the lower portion of Figure 3 (c), we show how this mapping serves to cluster learning outcomes together, and in upper part of this figure we show how this clustering imposes course-level prerequisites that lead to a curriculum dag. Thus, the solution space can be represented using three connected dags, one for the learning outcomes, one for the curriculum, and one that maps between the two. The goal in this work is to search over all possible mappings of learning outcomes to courses, in order to find ones that minimize the structural complexity of the resulting curriculum. Note that simply minimizing the total number of prerequisites in a curriculum does *not* necessarily produce a curriculum with the lowest structural complexity.

In order to demonstrate the usefulness of this work, consider the curricular design pattern shown in Figure 4 (a), commonly employed in electrical engineering programs for non-calculus-ready students. Notice the Circuits I course, a highly critical course in any electrical engineering curriculum, cannot be attempted, due to prerequisite constraints, until the fourth term in this curriculum. The structural complexity of this curricular design pattern is 71. An alternative curricular design pattern, shown in Figure 4 (b), can be created by the judicious rearrangement of a few learning outcomes among the courses in the design pattern, and it has show to more effectively serve non-calculus-ready students.[6] The work of finding a more effective curricular design pattern, in this case, was through the trail and error efforts of faculty over time. The goal here it to provide tools that would assist faculty in this work.

Optimal Learning Outcomes Assignment Problem

In the *Optimal Learning Outcomes Assignment* (OLOA) problem, the input is the collection of all course-level learning outcomes that must be attained by the learners in an academic program, or in some portion of the academic program, along with the prerequisite constraints among these learning outcomes (i.e., the order in which learning must occur). The goal is to assign these learning outcomes to the courses where they will be taught in such a way that the structural complexity of

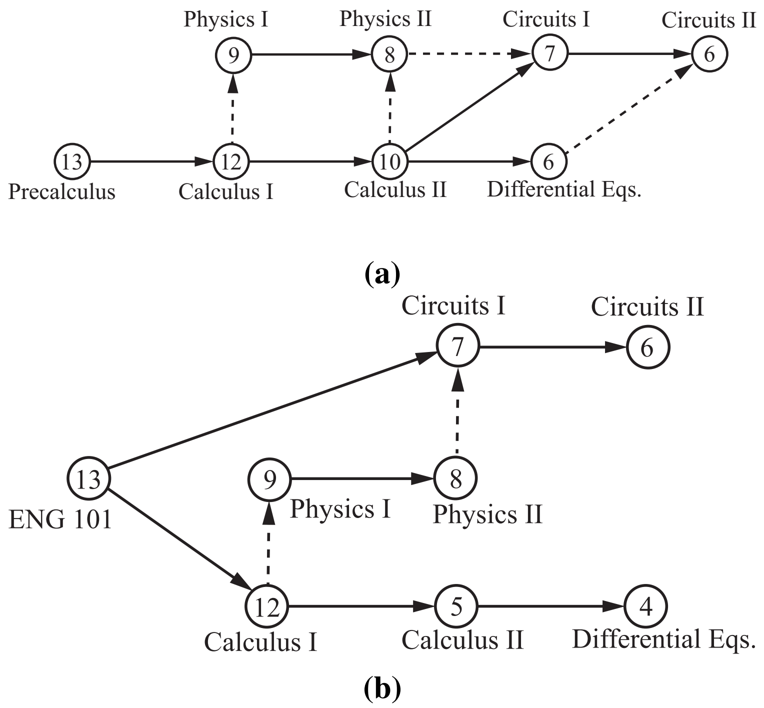

Figure 4: **(a)** A common curricular design patterns for attaining digital and analog circuit design learning outcomes in electrical engineering programs, assuming non-calculus-ready students. The solid directed edges denote prerequisites, and the dashed edges represent co-requisites. The structural complexity of each course is shown inside the vertex associated with that course. The total curricular complexity is 71. **(b)** A redesigned curricular design pattern involving the same learning outcomes, but with a lower curricular structural complexity of 64.

the resulting curriculum is minimized.

More formally, a *learning outcomes graph* can be represented as a dag $G_l = (V_l, E_l)$, where the vertices $v_{l_1}, \ldots, v_{l_m} \in V_l$ in this case denote m course-level learning outcomes, and the directed edges $(v_{l_i}, v_{l_j}) \in E_l$ correspond to the prerequisite arrangements between these learning outcomes. The assignment of learning outcomes to courses induces a *curriculum graph* $G_c = (V_c, E_c)$, where each vertex $v_{c_1}, \ldots, v_{c_n} \in V_c$ represents a course in curriculum c. For each edge $(v_{l_i}, v_{l_j}) \in E_l$, if learning outcome l_i is assigned to course c_p (denoted $l_i \to c_p$) and $l_j \to c_q$, then a directed edge (v_{c_p}, v_{c_q}) must be added to edge set E_c. That is, a prerequisite must be added to the curriculum graph.

Note that some learning outcomes-to-courses assignments can produce invalid curricula. In order for an assignment to be valid, it must be consistent. Specifically, if $(v_{l_i}, v_{l_j}) \in E_l$, $l_i \to c_p$, and $l_j \to c_q$, then for any other edge $(v_{l_k}, v_{l_t}) \in E_l$, if $l_k \to c_q$, then $l_t \not\to c_p$ (l_t cannot be assigned to c_p). In other words, all of the directed learning outcomes edges between any two courses must "point" in the same direction. Violating this condition will produce a cycle in the induced curriculum graph, leading to a curriculum that is impossible to complete. More generally, learning outcomes assignments that produce a cycle among any number of courses in the induced curriculum graph must be avoided.

By treating the assignment of a learning outcome to a course as a graph edge, an *outcomes mapping graph* can be constructed that shows how learning outcomes are mapped to the courses in a curriculum. Specifically, let $G = (V, E)$ denote a dag that contains G_c and G_l as subgraphs; that is, the vertex set $V = \{V_c \cup V_l\}$. The edges in G contain all edges in the two subgraphs, as well as an additional set of edges that specify how learning outcomes are mapped to courses; that is, $E = \{E_c \cup E_l \cup E_p\}$, where each edge $(v_{l_i}, v_{c_j}) \in E_p$ satisfies $v_{l_i} \in V_l$ and $v_{c_j} \in V_c$. That is, the mapping specified by E_p corresponds to the aforementioned curriculum map.

Now, consider a curriculum with m course-level learning outcomes, denoted $l_1, \ldots, l_m$, and let $h(l_i)$ denote the number of course contact hours required to attain learning outcome l_i. The objective, then, in the OLOA problem is to rearrange the m course-level learning outcomes among the n courses in a curriculum, so as to minimize the complexity of the resulting curriculum, while keeping the course contact hours in all courses within specified bounds. A formal definition of the OLOA problem is as follows:

Instance: A collection of learning outcomes $L = \{l_1, \ldots, l_m\}$ organized as a dag $G_l = (V_l, E_l)$, with $v_{l_1}, \ldots, v_{l_m} \in V_l$ and $(l_i, l_j) \in E_l$ if learning outcome l_j requires learning outcome l_i as prerequisite knowledge.
Question: Can L be partitioned into disjoint sets $C = c_1, \ldots, c_n$ such that $\alpha \leq \sum_{l_i \in c_j} h(l_i) \leq \beta$, for $j = 1, \ldots, n$, the curriculum graph induced by C is consistent, acyclic, and has minimal curricular complexity?

Theorem. *The decision version of the OLOA Problem is strongly $\mathcal{NP}$-complete.*

Proof: In the decision version of the OLOA problem, the question is whether or not a partition exists that yields a valid curriculum, where the number of hours in each course is in the range $[\alpha, \beta]$, and with curricular complexity at most k? Because it is possible to check in polynomial time whether a given partition is consistent, acyclic, within the range of allowable hours, and has curricular complexity at most k, OLOA is in $\mathcal{NP}$. The OLOA problem can be shown to be strongly $\mathcal{NP}$-hard through a reduction from the strongly $\mathcal{NP}$-complete 3-Partition problem (see[7]). Given an instance of the 3-Partition problem, consisting of $3t + 1$ positive integers $x_1, \ldots, x_{3t}$, and B, such that $\frac{B}{4} < x_j < \frac{B}{2}$ and $\sum_{j=1}^{3t} x_j = tB$, an instance of OLOA can be constructed in polynomial time as follows. The number of learning outcomes is set equal to $3t$, namely $l_1, \ldots, l_{3t}$, and the number of courses to t. Next, set the number of hours required for each learning outcome equal to $h(l_1) = x_1, \ldots, h(l_{3t}) = x_{3t}$, set $E_l = \emptyset$, and set $\alpha = \beta = B$. Then, any solution to the OLOA problem using these instances exists only if the integers $x_1, \ldots, x_{3t}$ can be partitioned into disjoint sets, each containing three elements, such that each set sums to B. $\square$

Given that the OLOA problem is strongly $\mathcal{NP}$-complete, the focus in the remainder of the paper will be on algorithms that yield good approximate solutions. However, one consequence of strong $\mathcal{NP}$-completeness is that unless $\mathcal{P} = \mathcal{NP}$, there is no fully polynomial-time approximation scheme for the OLOA problem (see[7]).

Integer Programming Algorithm

In this section, an approximation algorithm is proposed to address the OLOA problem. This approach uses 0-1 quadratic programming to create an assignment of m learning outcomes to n courses. That is, for a given outcomes mapping graph $G = (\{V_c \cup V_l\}, \{E_c \cup E_l \cup E_p\})$, the edges in E_p are mapped to a $m \times n$ binary-valued learning outcomes-to-courses assignment matrix as follows,

$$a_{ij} = \begin{cases} 1; & \text{if } (v_{l_i}, v_{c_j}) \in E_p, \\ 0; & \text{otherwise.} \end{cases}$$

The requisite edges in the resulting curriculum are specified using an $n \times n$ binary-valued requisite assignment matrix,

$$x_{ij} = \begin{cases} 1; & \text{if course } i \text{ is a prerequisite for course } j, \\ 0; & \text{otherwise.} \end{cases}$$

This is the adjacency matrix for the curriculum graph.

In addition, a $n \times k$ binary-valued courses-to-layers assignment matrix is defined as follows,

$$e_{ij} = \begin{cases} 1; & \text{if course } i \text{ is located in layer } j, \\ 0; & \text{otherwise,} \end{cases}$$

It is useful to recognize that any valid curriculum graph can be organized into layers (see[8]). A valid curriculum graph can be partitioned into k disjoint sets(or layers), labelled as $1, 2 \ldots k$. In such graph, no course prerequisite edges would appear within the same layer, while edges emanating from any layer terminate to its latter layer only.

Based on our studies of the characteristics of the structural complexities of curricula, we propose an objective function for the integer programming algorithm as followed.

$$\min\left(\zeta \sum_{j=1}^{n}\sum_{i=j}^{n} x_{ij} + \sum_{j=2}^{n}\sum_{i=1}^{j-1}(x_{ij} \cdot i) + \gamma \sum_{i=1}^{n}\sum_{j=1}^{k}(e_{ij} \cdot j)\right), \tag{1}$$

for constants ζ and γ, with $\zeta \gg n.$

Note that x_{ij}, the adjacency matrix for the curriculum graph, is constrained by the assignment matrix a_{ij} and the adjacency matrix for the learning outcomes graph. While the courses-to-layers assignment matrix e_{ij} is constrained by x_{ij}. To better understand the optimization problem specified by Equations (1), consider first the objective function given in Equation (1), which consists of three terms. Considering these terms from left to right, the first term forces the curriculum graph adjacency matrix to be upper triangular, thereby ensuring an acyclic graph, and therefore a valid curriculum. The second term tends to bias the non-zero entries in the curriculum graph adjacency matrix towards lower-valued i indices. This leads to connected components in the curriculum graph that are more compact. The third term in the objective function considers the impact to curricular complexity from the perspective of layers, as it is used to constrain the curriculum graph to span a smaller number of layers. As the number of layers is correlated with the maximal delay factor of curricular complexity, given a fixed number of courses, increasing the number of layers for a curriculum is likely to increase its value for curricular complexity. A reasonable value for k, the maximum number of allowable layers in the curriculum graph, can be determined *a priori* by inspecting the longest paths in the learning outcomes graph, as these dictate the minimum path lengths in the curriculum graph.

Results and Conclusions

An experiment was conducted to demonstrate the efficacy of the approximation algorithm described in the previous section. The integer programming algorithm is applied to the learning outcome graph underlying the electrical engineering curriculum shown in Figure 4 that starts with Precalculus. Figure 5 (a) shows this design pattern again, but with the courses relabeled as vertices $v_1, \ldots, v_8$. Figure 5 (b) shows the redesigned curriculum by running the approximation algorithm.

Comparing the two curricula in Figure 5, the complexity of the original curriculum is reduced from 71 to 42, by rearranging the learning outcomes. The new curriculum is about 41% less complex than the original curriculum pattern. Through the analysis of the longest path, it can be observed that the length of that path is 6 in the original curriculum, while in the redesigned pattern it is reduced to 4. This reduction in length indicates that students will require fewer terms to obtain the same set of learning outcomes in the redesigned curriculum pattern compared to the original pattern. Thus, the redesigned pattern provides more flexibility to students' study plans, which can potentially increase their graduation rates. In addition, the redesign process reduced the number of prerequisites and co-requisites from 10 to 8. A more compact curriculum allows the important learning outcomes to be taught in a timely manner, which could increase the efficiency of students' learning process.

Given the high complexity nature of engineering programs, it is important to explore various methods to help students with their studies. In this paper, we proposed a way from the perspective of rearranging the program-level learning outcomes from a curriculum, to reduce curricular complexity in the curricula design process. To assess the efficacy of this method, we formally defined the OLOA problem and proved the problem to be strongly $\mathcal{NP}$-complete. The proof indicates the direction of future research in this field. And our studies on the OLOA problem provide criteria to

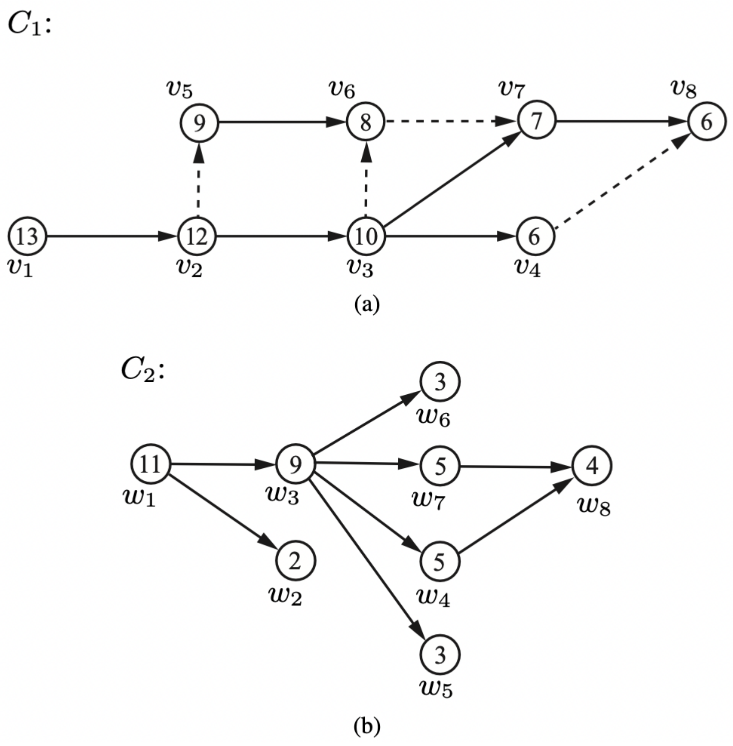

Figure 5: (a) The curricular design pattern from Figure 4, referred to as C_1 in this figure. The total curricular complexity of this design pattern is 71. (b) The curricular design pattern, referred to as C_2, found by rearranging the learning outcomes using the approximation algorithm described in this paper. The total curricular complexity of this design pattern is 42; that is, 41% less complex than the original design pattern shown in (a).

evaluate the performance of algorithms which aim at addressing this problem.

We proposed an approximation algorithm using integer programming techniques, which has demonstrated its efficacy on an electrical engineering curriculum. This algorithm can be applied to other disciplines as well. For future research, it is advisable to examine and incorporate other characteristics related to learning outcomes. For instance, in addition to the structure of the learning outcome graph, evaluating the difficulty level in teaching each learning outcome in a program can yield new metrics. By incorporating constraints on these metrics, a more balanced curriculum can be created. There could be various arrangements of learning outcomes that result in the same optimal curricular complexity. Given the diverse background information of students, further exploring the pool of optimal solutions could be another direction for future research.

References

[1] ABET. Criteria for Accrediting Engineering Programs: Effective for Reviews During the 2022–2023 Accreditation Cycle. 2022. `www.abet.org`.

[2] Edmund C. Short. A historical look at curriculum design. *Theory Into Practice*, 25(1):3–9, 1986.

[3] Grant Wiggins and Jay McTighe. *Understanding by Design*. Association for Supervision and Curriculum Development, Alexandria, VA, 2nd edition, 2005.

[4] Gregory L. Heileman, Chaouki T. Abdallah, Ahmad Slim, and Michael Hickman. Curricular analytics: A framework for quantifying the impact of curricular reforms and pedagogical innovations. *www.arXiv.org*, arXiv:1811.09676 [cs.CY], 2018. `arxiv.org/abs/1811.09676`.

[5] Gregory L. Heileman, William G. Thompson-Arjona, Hayden W. Free, and Orhan Abar. Does curricular complexity imply program quality? In *2019 ASEE Annual Conference & Exposition*, Tampa, FL, June 2019. ASEE Conferences. `peer.asee.org/32677`.

[6] Nathan W. Klingbeil and Anthony Bourne. The Wright State model for engineering mathematics education: Longitudinal impact on initially underprepared students. In *Proceedings of the 122nd ASEE Annual Conference & Exposition*, Seattle, WA, June 14–17, 2015.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 1979.

[8] Patrick Healy and Nikola S. Nikolov. How to layer a directed acyclic graph. In Petra Mutzel, Michael Jünger, and Sebastian Leipert, editors, *Graph Drawing 2001, Lecture Notes in Computer Science, vol 2265*, pages 16–30. Springer-Verlag, Berlin Hiedelberg, 2002.