Quantitative analysis of curricula coherence using directed graphs

Steffi Knorn [*], **Damiano Varagnolo** [**], **Kjell Staffas** [*], **Tobias Wrigstad** [***] and **Eva Fjällström** [****]

[*] *Dept. of Engineering Sciences, Uppsala University, Sweden*
[**] *Dept. of Engineering Cybernetics, Norwegian University of Science and Technology, Norway*
[***] *Dept. of Information Technology, Uppsala University, Sweden*
[****] *Dept. of Arts, Communication and Education, Luleå University of Technology, Sweden*

Abstract:
This paper investigates methods for quantitatively examining the connectivity and knowledge flow in a university program considering courses and concepts included in the program. The proposed method is expected to be useful to aid program design and inventory, and for communicating what concepts a course may rely on at a given point in the program. As a first step, we represent the university program as a directed graph with courses and concepts as nodes and connections between courses and concepts as directed edges. Then, we investigate the connectivity and the flow through the graph in order to gain insights into the structure of the program. We thus perform two investigations based on data collected from an engineering program at a Swedish university: a) how to represent (parts of) the university program as a graph (here called Directed Courses-Concepts Graph (DCCG)), and b) how to use graph theory tools to analyse the coherence and structure of the program.

Keywords: Courses Concepts Matrix (CCM), Courses Concepts Graph (CCG), university program design, graph theory, connectivity, max flow-min cut

1. INTRODUCTION

Creating and updating curricula is a complex and historically subjective task that involves creating and assessing proposals in different organizational and decision contexts. While there is little research on how different program designs affect students' engagement and knowledge development during their university education Ashwin [2014], Tight [2012], there is a significant body of literature on curriculum design, Wiles et al. [1989], including two established models, Gatawa [1990]: The *Objectives Model* starts by specifying the objectives or learning outcomes defined as measurable performances, and the *Process Model* starts by defining course content and specifying criteria to assess students' knowledge of these contents. Several variations on these models exist (e.g., Tyler's, Wheeler's, and Kerr's Models).

This is usually followed by dividing the required learning goals into smaller parts, often associated with different subjects and compartments, which translate into individual courses, for which often numerous complementary books and support materials exist, Slattery [2012], Connelly et al. [2008], Wiles [2008]. While this approach is practical for course design, it risks compartmentalizing learning, since it may fail to integrate courses, and may also fail in enabling students to understand how courses within a program are connected (which in turn has been shown to negatively impact learning Jones [2010]). Moreover, dividing a program into different courses is largely arbitrary and not necessarily focused on promoting students' acquisition of skills or knowledge.

In many cases, it is not certain what a specific course shall bring in form of decided (or intended) knowledge accordingly, and its connection to other courses and content of the program. Complicating matters further, courses may also be moved because for pragmatic reasons such as administration, they do not fit in the schedule from logistical reasons, or there is no teacher available. Even the profiles of the teachers can, consciously or subconsciously, change the curricula over time. Another problem is that a program board cannot have full control of all parts of a program, much less on how the course exams are done.

Further, higher education institutions are now investing more heavily in attempts to include research-based practices at all levels from teaching to program design and development. As pointed out in [Weaver et al., 2015, p. 6], this institution-level transformation is important, and should be reflected in opportune transformations of the university programs. However, the vagueness and lack of facility to objectively measure the goals of higher education may lead faculty members to realize and prioritize these goals based on their own interpretations Dee and Heineman [2016], Temple [2008]. Understanding how knowledge within a course or across courses in a program

⋆ Corresponding author: S. Knorn, steffi.knorn@signal.uu.se

interconnects can provide a useful basis for creating a structure and progression to support students' learning.

A renowned strategy for better understanding and designing the connection between different parts in a university program is the so-called *black-box* approach to the sequencing of a curriculum [Crawley et al., 2014]. This development tool has been proposed within the Conceiving, Designing, Implementing, and Operating (CDIO) standard to the management of university programs, and consists in representing every course within a program as a set of inputs (e.g., pre-required knowledge and skills) and outputs (e.g., contributions to the final learning outcomes). Coupling this information from all courses is expected to enable discussions, make connections (or their lack of) visible, provide an overview of the program, and eventually serve as a basis for both planning and improving. However, this tool is still qualitative, and does not provide quantitative indications that are not subject to personal interpretations. Another approach, presented in Aldrich [2015], analysed the connections between courses according to the curriculum structure in order to analyze coherence and structure within a university program. However, the important questions *why* courses are prerequisites for other courses and *what is learned* in the courses was not considered.

In fact, to the best of our knowledge and experience as teachers at university level, *quantitative* tools are not used to steer the development process or to check for design flaws involuntarily caused by the lack of holistic knowledge on the structure of the learning flows within the curricula. Hence, intuitively, curricula design and modification processes may benefit from data-driven tools that strive for giving objective and context-independent information. In other words, evidence-based information infrastructures can support taking subjective but informed decisions in the program board meetings, written exchanges, and face-to-face discussions among peers. This is specially important creating, expanding or substantially modifying a program in terms of its curriculum, pedagogy, and/or learning outcomes. To be effective, these actions should be supported with up-to-date and objective information, e.g., on labor market conditions and available resources at the institution [Posey and Pitter, 2012]. It is also important to recognize that any type of data-driven evidence should always be interpreted within the organizational and decision contexts to avoid misinterpretations [1], as clearly remarked for example by Dee and Heineman [2016].

The method considered in this paper serves the needs above. Summarizing, its derivation followed this intuition: at the university level, and specially for engineering disciplines, Intended Learning Outcomes (ILOs) often correspond to mastering individual concepts and procedural knowledge (e.g., understanding linearity and linear independence, or being able to program a microcontroller). Ideally, the sequence of courses in a university program reflects a learning flow starting from basic knowledge and leading to increasingly complex understanding. In other words, courses build students' competences by adding layers of knowledge while laddering on the previously acquired concepts and prerequisites.

The flow of courses can thus be seen as a logical flow of concepts and prerequisites, and it is meaningful to capture these through opportune graphs. Framing a program as an opportune graph allows then to use concepts from graph theory to analyze the properties of these curricula, so that the problem of analyzing university programs can be cast in terms of graph analysis.

In this manuscript we, therefore:

- propose a strategy to collect information on the structure of the program from the individual teachers that is amenable to quantitative analysis;
- propose algorithms to transform this raw information into directed, bipartite graphs, which involve the courses of the program as well as key learning contents;
- discuss how classical graph-theoretical connectivity analysis can be interpreted for the pedagogical purpose of inferring potential flaws in a given university program;
- apply this method to a real-world case at a Swedish university and analyze the consequent numerical results.

Section 2 describes the tools for collecting and representing quantitative information about a generic university program. Section 3 discusses how classical connectivity results from graph theory can be interpreted and applied in our university programs analysis context. Section 4 reports and examines the results obtained from field applications of the proposed method. Finally, Section 5 presents some concluding remarks and suggests some future research and development efforts.

2. THE DIRECTED COURSES-CONCEPTS MATRIX AND THE DIRECTED COURSES-CONCEPTS GRAPH

To quantitatively evaluate and analyze the structure of university programs, we exploit the previously developed intuition that courses within a program are connected through a flow of concepts and skills. For instance, while some concepts are prerequisites for understanding and successfully following a course, other concepts should have been learned and understood when passing a course. To highlight the connections among courses, we follow a procedure based on executing two separate steps: *acquire the data*, as described in Section 2.1, and *visualise and represent the data*, as described in Section 2.2.

2.1 Data acquisition: Directed Courses-Concepts Matrix

In its simplest form, a Courses-Concepts Matrix (CCM) is a table where element (k, j) quantifies how relevant the concept k is for course j on a predefined scale. A simple scale might use "0" = not relevant, "1" = relevant but not central, and "2" = very relevant / central for the course. See Fjällström et al. [2018] for details.

A main drawback of this method is the lack of insight into why a given concepts is relevant for a given course. Hence,

[1] E.g., the data may be just not used, or misinterpreted, questioned, or even become a source of complaint when stakeholders perceive that they have not been involved in the information collection and processing steps Schmidtlein [1999].

	1DT051 information technology	1DL201 data structures	1DT093 computer architecture
recursion	1 T	2 T	
divide & conquer	1 T	1 T	
induction	1 T	1 R	
data structures	1 T	2 T	
trees	1 T	2 T	
lists	1 R	2 T	
graphs		1 T	
arrays	1 R	1 T	
hashtables		1 T	2 T

Fig. 1. A small part of a Directed Courses-Concepts Matrix taken from the field case of Computer and Information Engineering, Uppsala University, Sweden. (T = 'taught' in the course, R = 'required' in the course)

in this work, the CCM is extended by also collecting information on whether the concept is relevant as a prerequisite for the course or is a relevant/important learning goal of the course. Collecting this type of information allows more detailed analyses; to enable these, one may let the CCM have two columns for each course: one allocated for weights to quantify the relevance of prerequisite concepts (e.g., the learning levels that students should ideally have to be able to follow fruitfully the course), and the other one allocated for weights of concepts taught or developed in that course (e.g., the learning levels that students should ideally reach about that concept after having passed the course). Alternatively, one may indicate with an opportune symbol whether a specific concept is required or taught by a specific course.

In our field study, we created the Directed Courses-Concepts Matrix (DCCM) for the seven central programming courses and 111 concepts in the program Computer and Information Engineering at Uppsala University. A small part of it is shown in Figure 1 and more detailed results will be discussed in Section 4.

Several possibilities exist to collect the necessary information to build the DCCM for a program. One option, which is also followed in this paper, is to interview teachers in the program about their courses. This method has obvious drawbacks as it highly depends on the engagement of teachers into their course and willingness to contribute to such a project but also on their interpretation of the scale. In order to minimize the subjectivity of the provided information, clear instructions and explanations should be provided to the teachers in order to establish a common understanding of the scale. Further, it should be clarified that, if possible, the information on a course should be general for this course and not mirror small adaptations in a particular academic year.

Data can also be collected by interviewing students in the program. Here, a tradeoff between asking students soon enough to avoid them forgetting aspects of the course and asking them late enough to allow for reflections is important. In order to minimize subjectivity of individual opinions, the average over several students' opinions might be used or combined with the input from the teacher. Further, if suitable, exams and exam questions might also be analysed to distinguish between what teachers and students perceive as relevant and what is actually required to pass the course.

2.2 Data representation: Directed Courses-Concepts Graph

After obtaining the DCCM described above, the program can be represented as a directed, weighted, bipartite graph, denoted Directed Courses-Concepts Graph (DCCG). The two sets of nodes in this graph correspond to the courses j and the concepts k within the program. Element (k, j) in the CCM corresponds to the weight of the edge between the concept node k and the course node j. Edges that are directed from concept nodes to a course node indicate that the concepts are required for that course, while edges that are directed from course nodes to concept nodes indicate that the concepts are taught by that course.

Intuitively, the properties of a university program (e.g., its structure, the relations and the relevance of the courses and concepts in a program, the existence of potential flaws in its design) should translate into opportune topological properties of the DCCG. If this intuition is true, then the problem of quantitatively analyzing a university program can be cast as the problem of analyzing the opportune graph. For example, combining the additional information on what are requirements and what are course outcomes can be used to, e.g., discover when early courses treat a given concept as prior knowledge despite it being only introduced in a later course. The problem of understanding what can actually be inferred about a university program through analyses of its DCCG is discussed in Section 3 and results for a field study at Uppsala University (UU) are discussed in Section 4.

3. CONNECTIVITY AND FLOW ANALYSIS

In this section we assume to have collected enough information so that, for a given university program or parts of it, both the relative DCCM described in Section 2.1 and the corresponding DCCG in Section 2.2 have been compiled. Due to the special structure of these tools (i.e., a matrix and a graph), one can cast the problem of analyzing the properties of the program into the problem of analyzing the properties of the corresponding matrix or graph by means of well known and established tools from matrix and graph theory. Our purpose is then to discuss what these established tools say about potential structural problems of the represented programs.

3.1 Connectivity

Intuitively, an undirected graph is connected if there exists a path between each pair of nodes. Otherwise it is disconnected. For directed graphs, three different definitions exist. First, consider ignoring the direction of all edges, i.e., replacing the directed graph with an undirected graph. If the corresponding undirected graph is connected, the original directed graph is weakly connected. Further, the directed graph is connected if for each pair of nodes i and j, there exists at least a directed path from node i to j or from j to i. Finally, if both directed paths from node i to j and from j to i exist for all pairs of nodes i and j, the graph is strongly connected. Since DCCG can be interpreted as a description of a knowledge

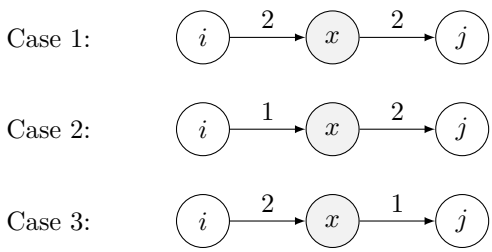

Fig. 2. Examples of basic DCCGs, where i and j refer to courses, and x refers to a concept.

network, we expect it not to be strongly connected but at most connected. Further, if a DCCG is disconnected or only weakly connected, it reveals that different parts of the university program have no overlap or connection or that mismatches exist between required prior knowledge and previously taught concepts.

3.2 Minimal cut

In relation to the concept of connectivity, a natural question that arises is to understand which edges or nodes are essential to maintain connectivity, i.e., which elements must not be removed to avoid making the graph disconnected. Such sets of edges or nodes are denoted edge or vertex cuts, respectively. Further, the minimal cut of a graph describes how sensible a graph is to loose its connectivity. Both the size of the minimal cut (amount of edges or vertexes to be removed) as well as the edges or nodes included in the minimal cut offer important insights into the structure of the graph.

Analysing the minimal cut of a DCCG can give interesting insights into the program structure and its vulnerabilities. For instance, the minimal vertex cut will list the courses or concepts whose inclusion in the program is crucial to connect different areas or aspects in the program. Also, the minimal edge cut will reveal which conveyance of certain concepts in certain courses is crucial to maintain connectivity on the program. In other words, it will indicate which concepts in which courses connect different areas within a program.

3.3 Network flow

The weight of an edge in a network can be interpreted as its capacity to carry a physical flow. Interpreting every edge of a network as such a capacity, it is natural to ask how many units of flow can be transported from one part of the network to another. Defining thus at least one source node and at least one sink node, it is possible to compute the maximal admissible flow that can be carried between these nodes through well-known algorithms, e.g., Diestel [2005]. Under this maximal flow situation it is possible that some edges carry less flow than their maximum admittable ones.

The residual capacity of an edge is then the difference between its natural capacity versus its usage under maximal network flow situations. Also, note that finding the maximal flow of a network is equivalent to finding an edge cut of minimal capacity that would separate the sink from the source.

In the pedagogical context of considering the DCCG, we can interpret the weights 0, 1 and 2 as capacities describing two specific phenomena. Referring to Figure 2, possible interpretations are:

- when the edge is from a concept x to a course j, how much the prior understanding of the required concept x contributes to learn the ILOs of the course j;
- when the edge is from a course i to a concept x, how much the course i contributes to teach / facilitate the understanding of the concept x.

This gives rise to a natural question: how can analysing the maximal network flow associated to a DCCG be helpful in understanding the structure of a program, its shortcomings, bottlenecks and redundancies?

First of all, to enable performing network max-flow analyses, at least one source and one sink node must be defined. For this, we propose to create two additional nodes: "0", as a global source, and "∞", as a global sink of the network flow, so that:

- node 0 symbolises the prior knowledge that students are expected to have before starting a certain program. Node 0 connects then with infinite capacity to all concepts that are considered a required prior knowledge, e.g., from high school education. Note that, if for a given student or student cohort it is, however, known that their prior knowledge of some required concept is limited or lacking, the capacity of the corresponding edge from 0 to this concept node can be lowered to analyse the consequences of this shortcoming;
- node ∞ represents the final knowledge that students are expected to have after finishing a certain program. All those concepts whose understanding is included or required for reaching the goals of the program should thus be connected to node ∞ with edges of infinite capacity.

We expect that the decision about which concepts shall be connected to nodes 0 and ∞ may require the teachers and boards of the various programs extensively discussing the program structure.

Assume then to have added to an existing DCCG these fictitious source and sink nodes, their corresponding edges, and to have computed the maximal and the residual flows of the network. Some interesting cases may then happen: First, consider Case 1 in Figure 2, where both edges have the same capacity. Hence, the maximal flow is such that everything entering in concept x can also flow to course j. This can be interpreted as a well aligned structure, where students are enabled in course i to build up an appropriate knowledge about x, and then use it in course j.

In contrast, Cases 2 and 3 in Figure 2 might both reveal some mismatch in the program. In Case 2, indeed, the overall maximal flow from i to j is 1, since concept x is only taught with weight 1 in course i even though it is required with weight 2 by the following course j. Hence the residual flow of the edge from x to j is 1, and a plausible interpretation of this is that students may not be as well prepared for taking course j as expected by the teacher of this course. In Case 3, instead, it is the link from i to x

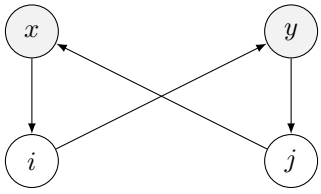

Fig. 3. Example of a cycle in a DCCGs, where i and j refer to courses, while x and y refer to concepts. The weights of the edges are omitted for simplicity.

that has under the max-flow regime a residual capacity 1. A plausible interpretation of this is that not all the effort that is put during course i into teaching x is required for the following-up course j. This indicates "wiggle room" for shifting teaching efforts, e.g., reducing this potentially "unnecessary" effort during course i could free resources e.g., to counter a mismatch as in Case 2.

In extreme cases there may be edges with maximal residual flows, i.e., edges that are completely unused under maximal network flow regimes. Our expection is that these events indicate grave mismatches or inconsistencies within the analysed program. This might include concepts that are neither taught in previous courses nor can be considered knowledge that students should have before starting the program, but are nevertheless required for some course. This situation is especially critical, since the lacking of this required knowledge may hinder the learning of new concepts. The event may also indicate situations where the teaching includes concepts that are neither required in later courses nor considered desired program goals (i.e., unnecessary material).

Finally, minimal capacity cut shows where the program is least robust, in the sense that it shows which concepts and courses are key for reaching the overall program goal.

3.4 Existence of cycles

Another pedagogically interesting analysis of a DCCG is related to detecting cycles in the graph. For example, consider the situation in Figure 3: here course i has concept x as a required knowledge, and prepares students to course j by teaching concept y. The situation is though symmetric, since j has concept y as a required knowledge, and prepares students to course i by teaching concept x. Thus as soon as i and j are not taught in the same learning period, students will not be prepared to take the first of the courses being taught (something that theoretically will also affect their understanding, eventually inficiating also the attendance to the second one). And even if i and j are instead taught in the same learning period, this logical fallacy can be resolved only through a great care by the teachers of i and j in designing their own courses so that the understanding and usage of concepts x and y happen in parallel in a well-coordinated fashion.

Generalizing, moreover, plotting a DCCG so that the course and concept nodes follow a temporal order (as was the case in the field-example from UU in Section 4) makes every link that points "backwards" highlight some program inconsistency (i.e., a situation where students will not be prepared to take a certain course because the required concepts are being taught in a consequent learning period). Note however that if no cycles are present, though, this specific problem may be resolved by opportunely changing the temporal order of when the various courses are given.

4. RESULTS

We gathered data for the program Computer and Information Engineering at UU, Sweden, by asking teachers to allocate weights of the scale $\{0, 1, 2\}$ for the concepts in their course in the program according to the method described in Sections 2.1. This included seven core programming courses and a total of 111 concepts. Since the corresponding DCCG is too large to be shown here, only a small part of it is shown in Figure 4.

Analysing this DCCG reveal several interesting insights. First of all, analysing the max flow of the graph allows to understand mismatches in effort in the sense of spending much more time or effort in teaching a concept, than what is required in few or no courses with low weights, or vice versa. For instance, consider "hashtables", which is taught in three courses, 1DL201, 1DT093 and 1DL221 with accumulated weight 4, but only required in one single course, 1DT096, with weight 1. In contrast, the concept "induction" is only taught in one early course, 1DT051, with weight 1 but required for three following courses with various weights and further considered an intended learning output of the program. Both cases can be found by analysing the redundancies in the DCCG under max flow. (Links with maximal flow in Figure 4 are shown in green and links with redundancies are shown in blue.)

Further, analysing cycles in the graph allowed the extraction of mismatches in time (marked as red arrows in Figure 4). For instance, between courses, where concepts (e.g. "lists" and "arrays") are assumed prior knowledge, or at least a common understanding of them, and are hence required in an early course (e.g. 1DT093) but are taught in a later course (e.g. 1DT051). Either, teaching these concepts in the later courses is redundant since the teacher of the first course correctly assumed that students would know these concepts from earlier education as for instance high school; or students are not sufficiently prepared for the early course due to a lack of knowledge. It may also be possible that the courses require or teach the concept on different levels in Bloom's taxonomy. In this case, no mismatch might be present. However, in order to understand those issues and to avoid mismatches, teachers need to collaborate and/or exchange more detailed information. In any case, the analysis of the corresponding DCCG allows to identify which aspects need to be discussed or maybe even changed in the course and program organisation.

5. CONCLUSIONS

In this paper, we proposed a method to analyse quantitative data about which concepts are relevant for which courses in a university program and the connections between them, distinguishing between concepts, that are required for a course, and concepts, that are taught in a course. We showed how this information can be described by a DCCM and the corresponding DCCG. Their analysis can be used to reveal mismatches and redundancies in

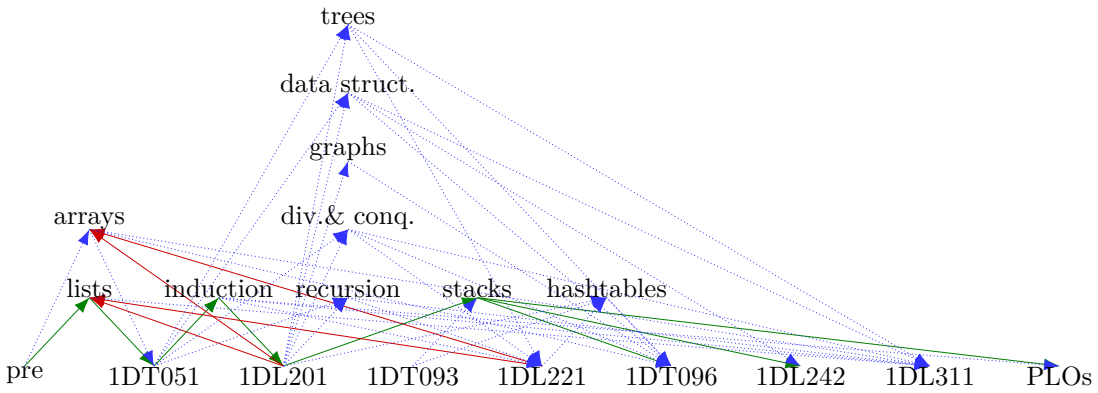

Fig. 4. Part of the DCCG for Computer and Information Engineering at UU showing temporal mismatches as red arrows, unused/redundant connections as blue arrows and remaining (well matched) connections are green arrows. (Note that the max-flow analysis involved all concepts but only 10 are shown here.)

the program. To illustrate the usefulness of the proposed method, data from a field study at UU were presented.

Additionally, input from students might be of use in the process of establishing concepts for courses in two major ways. Firstly, to determine whether the concepts identified by the teachers match the experiences of the students. Secondly, the concepts identified by the students function as feedback for teachers and program boards on how courses are experienced by students. This is valuable input for course and program development. Large discrepancies indicate that the course misses the target, which could be an issue in ensuring development and progression of an intended learning curve.

Comparing DCCGs of similar programs in different institutions might also be used to understand structural differences and similarities and foster teaching collaborations.

Several interesting future research topics arise from this work: First, suitable methods should be developed to visualise and display the graph structure and highlight the obtained results such as flows, cycles and discrepancies in a suitable way. Even though our preliminary experience in communicating our method to students, boards, administrators and fellow teachers revealed great interest and readiness to adopt the new method, its success will depend on whether the information can be presented in a useful and understandable way for all stakeholders.

Further, one should also investigate how the method can be adapted to other study areas. For example, in order to capture other aspects of learning, not only concepts but also facts and procedures should be considered.

REFERENCES

P. R. Aldrich. The curriculum prerequisite network: Modeling the curriculum as a complex system. *Biochemistry and Molecular Biology Education*, 43(3), 2015.

P. Ashwin. Knowledge, curriculum and student understanding in higher education. *Higher Education*, 67(2): 123–126, 2014.

F. M. Connelly, M. F. He, and J. Phillion. *The Sage handbook of curriculum and instruction*. Sage, 2008.

E. F Crawley, J. Malmqvist, S. Östlund, D. R. Brodeur, and K. Edström. The CDIO approach. In *Rethinking engineering education*, pages 11–45. Springer, 2014.

J. R. Dee and W. A. Heineman. Understanding the organizational context of academic program development. *New Directions for Institutional Research*, 2015(168):9–35, 2016.

R. Diestel. *Graph Theory*. Springer-Verlag Heidelberg, New York, 2005.

E. Fjällström, S. Knorn, K. Staffas, and D. Varagnolo. Developing concept inventory tests for electrical engineering: extractable information, early results, and learned lessons. In *Proceedings of the UK Automatic Control Conference*, 2018.

B. S. M. Gatawa. *The politics of the school curriculum: An introduction*. College press, 1990.

C. Jones. Interdisciplinary approach-advantages, disadvantages, and the future benefits of interdisciplinary studies. *Essai*, 7(1):26, 2010.

J. Posey and G. W. Pitter. Supporting the provost and academic vice president. *The handbook of institutional research*, pages 145–164, 2012.

F. A. Schmidtlein. Common assumptions about organizations that mislead institutional researchers and their clients. *Research in Higher Education*, 40(5):571–587, 1999.

P. Slattery. *Curriculum development in the postmodern era: Teaching and learning in an age of accountability*. Routledge, 2012.

P. Temple. The integrative university: Why university management is different. *Perspectives*, 12(4):99–102, 2008.

M. Tight. *Researching higher education*. McGraw-Hill Education (UK), 2012.

G. C. Weaver, W. D. Burgess, A. L. Childress, and L. Slakey. *Transforming institutions: undergraduate STEM education for the 21st century*. Purdue University Press, 2015.

J. Wiles. *Leading curriculum development*. Corwin Press, 2008.

J. Wiles, J. Bondi, and H. Guo. *Curriculum development: A guide to practice*. Merrill Publishing Company, 1989.