

Network models for mapping educational data

Karen E. Willcox¹ and Luwen Huang²

¹Department of Aeronautics and Astronautics, MIT, Cambridge, MA 02139

²The Mapping Lab, MIT, Cambridge, MA 02139

Abstract

Educational mapping is the process of analyzing an educational system to identify entities, relationships and attributes. Current mapping processes in education typically represent data in forms that do not support scalable learning analytics. To overcome this limitation, this paper proposes a network modeling approach to educational mapping. The paper presents network models for educational use cases, with concrete examples in curriculum mapping, accreditation mapping and concept mapping. Illustrative examples demonstrate how the formal modeling approach enables visualization and learning analytics. The analysis provides insight into learning pathways, supporting design of adaptive learning systems. It also permits gap analysis of curriculum coverage, supporting curricular design at scales ranging from an entire institution to an individual course.

Introduction

Education contains a wealth of linked data whose key value lies in its connections. Existing processes underscore the value in exploring these relationships at a variety of scales: the mapping of prerequisite linkages across courses can identify gaps, overlaps and pathways in a curriculum redesign [13, 2, 16], the linking of learning outcomes to educational resources is a necessary ingredient in designing adaptive learning systems, and the mapping of concepts in a concept map is a valuable exercise for instructional designers [22]. By analyzing the linkages within connected data, we can better design educational experiences [9, 16].

Educational mapping is the process of analyzing an educational system to identify entities, relationships and attributes. Current mapping processes in education (such as curriculum mapping and concept mapping) typically represent data in forms that do not support scalable analysis. In particular, highly-connected data are often represented in implicit forms where relationships within the data appear as flattened attributes of the entities they link [12]. This lack of a *first-class* representation results in loss of information and forces ad-hoc mechanisms to analyze connections in the system [1]¹. While acceptable for the one-off study, this manual approach does not scale to large or dynamic data sets, nor does it provide a structured foundation for visualization and analytics.

In this paper, we provide a structured scalable model on which to conduct educational mapping. We propose a network-based approach to modeling highly-connected educational data. Network models are used in many fields to model entities and the relationships between them. Examples include social and organizational networks [8, 10, 11, 15], biological networks [3, 18], and

¹For an introductory background on the notion of first-class citizenship in computer science, we refer the reader to [19].

Corresponding

author
Karen
E.
Will-
cox
kwillcox@mit.edu

Published
by
Cam-
bridge
University
Press
under
the
Cre-
ative
Com-
mons
Attribution
li-
cense
(<http://creativecommons.org/licenses/by/3.0/>)

©
The
Au-
thor(s)

Des.
Sci.
journals.cambridge.org/dsj
DOI

the **Design Society**
a worldwide community

 **CAMBRIDGE**
UNIVERSITY PRESS

transportation networks [4]. A primary strength of network models lies in their ability to explicitly represent relationships as first-class objects instead of as derived properties of other objects in the model. Here, we make the case that such a modeling approach is essential for representing, visualizing and analyzing educational data at scale. Scalable modeling is particularly important if the promise of learning analytics and educational analytics is to be fully unlocked [7, 21]. To motivate from concrete examples, we consider three use cases highly relevant in educational analytics—curriculum mapping, accreditation mapping and concept mapping—and we present an approach to formally model and analyze the corresponding data sets as a network.

Network models for education

In this section we first introduce some basic concepts of network modeling and graph theory. We then present three educational network models: a curriculum mapping model, an accreditation mapping model, and a concept mapping model. For each, we define the elements of the network model and discuss how the tools of graph theory can provide analysis and design of educational structures at different scales.

Basic concepts of network modeling

A graph structure G , notionally portrayed in Figure 1, consists of a set of *vertices* and *edges* between vertices. Vertices represent entities in a system, and edges between vertices represent relationships between entities. An edge is assigned text and numeric attributes, such as name, directionality, cost and weighting of the relationship. A vertex is assigned text and numeric attributes that represent information on the entity, including its name and other relevant properties. For a *directed edge*, the relationship applies only in one direction—the direction of the edge matters and is indicated visually with an arrow. An *undirected edge* is bidirectional—the relationship applies in both directions. Visually, this is typically indicated by no arrow. The words *graph* and *network* are often used interchangeably; in this paper we will use the term *graph* when discussing the theory and methods of graph structures and the term *network* to describe a situational structure with real world data representation.

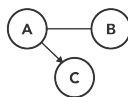


Figure 1: A graph structure G with three vertices and two edges. The edge from vertex A to vertex C is directed, while the edge between vertex A and vertex B is undirected.

Several basic concepts of graph theory will be useful in analyzing our educational network models. We summarize those concepts here (with a minimal level of mathematical detail) and refer the reader to the textbooks [5, 6, 23] for more detail.²

²Wikipedia also has a useful “Glossary of graph theory terms” https://en.wikipedia.org/wiki/Glossary_of_graph_theory_terms.

The *degree* of a vertex is defined as the number of edges between the vertex and other vertices in the graph. The degree distribution of the graph $P(k)$ is the fraction of vertices in the graph with degree k . When edges are directed, the analysis of a vertex's degree accounts for directionality: the *indegree* of a vertex is defined as the number of incoming directed edges from other vertices, and the *outdegree* of a vertex is defined as the number of outgoing directed edges to other vertices.

We will sometimes be interested in analyzing just a portion of the network model. The term *subgraph* of a graph G is another graph formed from a subset of the vertices and edges of G . In some cases, we will analyze the pathways associated with a directed subgraph. A *topological sort* of this subgraph results in an ordering of its vertices where for every directed edge from a vertex A to a vertex B , A is ordered before B . We can visually draw this subgraph with vertices arranged according to increasing path length from a source vertex, as shown notionally in Figure 2. We can then assign a rank to each vertex in the subgraph, where the rank of vertex V is defined to be the length of the longest path from the source vertex to vertex V .

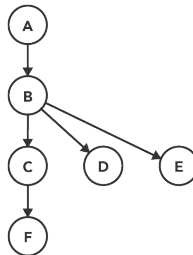


Figure 2: A directed graph with vertices arranged by assigned rank. Relative to the source vertex (Vertex A): Vertex A has a rank of 0; Vertex B has a rank of 1; Vertex C, Vertex D and Vertex E have a rank of 2; and Vertex F has a rank of 3.

The curriculum mapping model

We define a curriculum mapping network model as follows. We first define the different types of entities that reflect organization of the curriculum. In our basic curriculum model, our entities have type Course, Department or Institution. We then define all the entities in the curriculum model: the courses, the departments, and the institution. Each entity of these various types is modelled as a vertex in the curriculum network model. They are assigned attributes, which might include their name, course number, schedule information, website listing, etc. If the curriculum comprises n courses across m departments and a single institution, then the total number of vertices in the network model is $n + m + 1$.

Next, we define different types of relationships among the entities. A basic curriculum model has three types of relationships: *has-parent-of*, *has-prerequisite-of* and *has-corequisite-of*. The *has-parent-of* relationship is a directed relationship that specifies organizational hierarchy in the curriculum. For example, we define *has-parent-of* relationships from a course to its parent department, and from a department to an institution. These relationships are modelled as directed edges between the appropriate vertices in the curriculum network model. The *has-prerequisite-of* and *has-corequisite-of* relationships

are directed; they exist between courses to define prerequisite and corequisite relationships. Again, these relationships are modelled as directed edges between the appropriate vertices in the curriculum network model. If each course and each department has a single parent, there are $n+m$ parent grouping relationships. With r prerequisite relationships among classes and q corequisite relationships, there is a total of $n + m + r + q$ directed edges in the network model.

Figure 3 shows the curriculum model ontology corresponding to this network model. In the models presented in this paper, a course is the most granular entity of a curriculum model, but one could also define a more granular entity type, such as module (or some other unit of learning smaller than a full course). Similarly, one could introduce other units of organization, such as degree program, school, etc.

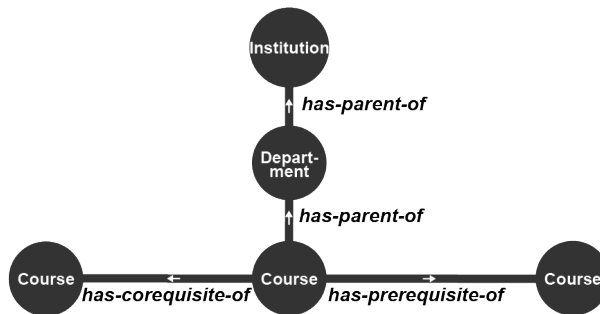


Figure 3: A curriculum mapping model ontology.

This modeling approach can be used to model other, more complicated aspects of a curriculum. For example, sometimes multiple courses can fulfil a given prerequisite requirement. In this case, the basic *has-prerequisite-of* relationship does not capture the “or” nature of the requirement; however, the network model can be extended to represent this situation. Consider a course A that requires the student to complete ℓ prerequisite courses chosen from a set $C = \{c_1, c_2, \dots, c_s\}$, where each c_i is a course and s is the number of courses in the set of possibilities. Note that by definition, $s \geq \ell$, and if $s = \ell$, we are in the simple case where there are no alternative options for a given prerequisite. C can then be modelled by a single vertex labeled OR. We attach a *required-number* attribute to the vertex with value ℓ . The vertex representing course A is connected to the OR vertex via a *has-prerequisite-of* relationship. The OR vertex is further connected to the s vertices representing the courses in the set C , via *can-be* relationships. This modeling is illustrated in Figure 4.

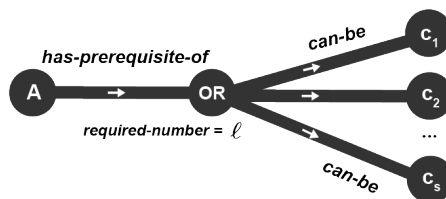


Figure 4: Capturing the situation of alternate prerequisite requirements by introducing an OR vertex and *can-be* relationship to the network model.

With the curriculum network model defined, we can now use the tools of graph theory to analyze and visualize the curriculum. A reachability analysis reveals pathways through the curriculum, i.e., we can identify all the class nodes in a curriculum network model that are reachable from a given class node, thus showing all the downstream courses that flow from an upstream prerequisite. Similarly, one could apply topological sorting to subgraphs of the curriculum network, to identify entire prerequisite chains of a course.

To visualize the curriculum network, we could simply draw all entities and relationships as nodes and edges, as many graph visualizations do. However, we find that for the purposes of curriculum design and institutional analysis, it is more useful to visualize courses grouped into their respective departments. Therefore, in our visualization application, we draw nodes for only courses, and edges between nodes for only *has-prerequisite-of* and *has-corequisite-of* relationships. Nodes are visually grouped into clusters, driven by the existence of *has-parent-of* relationships. The types of entities and relationships for the curriculum network model and its visualization can be seen in Table 1.

Table 1: Our educational network models are defined by different types of entities and relationships. Each model has a tailored visualization strategy.

Model	Entities	Visualized nodes	Relationships	Visualized edges
Curriculum mapping	Course Department Institution	Course	<i>has-parent-of</i> <i>has-prerequisite-of</i> <i>has-corequisite-of</i> <i>can-be</i>	<i>has-prerequisite-of</i> <i>has-corequisite-of</i>
Accreditation mapping	Outcome Course Program	Outcome Course	<i>has-parent-of</i> <i>addresses</i>	<i>addresses</i>
Concept mapping	Outcome Concept Module Course	Concept Outcome	<i>has-parent-of</i> <i>leads-to</i> <i>addresses</i>	<i>leads-to</i> <i>addresses</i>

The accreditation mapping model

Accreditation mapping is the process of mapping learning evidence to accreditation outcomes in order to show how accreditation outcomes are met. As well as supporting program evaluation, accreditation mapping is used in curriculum redesign [14, 17].

We define an accreditation mapping network model as follows. We define four different types of entities: Outcome, Course, Group and Program. Each entity of these various types is modelled as a vertex in the accreditation network model. We define two types of relationships: *has-parent-of* and *addresses*. As in the curriculum model, the *has-parent-of* relationship is a directed relationship that specifies organizational hierarchy; in this case it relates courses to a group, and groups to a program. The *addresses* relationship is a directed relationship that indicates that a course addresses an outcome. These relationships are modelled

as directed edges between the appropriate vertices in the accreditation network model. We create a directed edge of *has-parent-of* type from Course A to Group X if Course A belongs to Group X (examples of groups are Elective Courses, Core Courses, Capstone Courses, etc.). Similarly, we create a directed edge of *addresses* type from Course A to Outcome T if Course A addresses Outcome T. The *addresses* edges are assigned a weighting to indicate how strongly a course addresses an outcome. Figure 5 shows this accreditation mapping model ontology. Note that one could easily introduce additional types of entities and/or relationships to the model as desired.

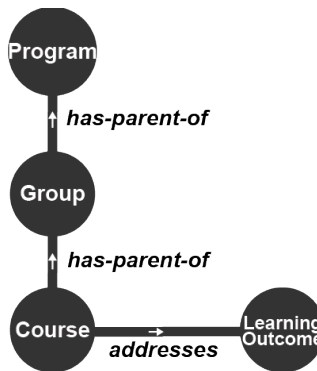


Figure 5: The accreditation mapping model ontology.

With the accreditation network model defined, graph analytics reveal how program structure relates to accreditation requirements. The indegree of an outcome vertex defines how many courses contribute to addressing that outcome. For a given course vertex, the outdegree corresponding to edges of type *addresses* defines the number of outcomes addressed by that course. Analyzing the distribution of indegree and outdegree over the network model gives insight into a program’s strength of coverage across outcomes.

As with the curriculum model, a tailored graph visualization is more useful than a generic graph visualization. In our visualization application, we draw nodes for only courses and outcomes. We visualize courses grouped into their respective groups and programs, driven by the existence of *has-parent-of* relationships. We draw edges between nodes for only *addresses* relationships. The types of entities and relationships for the accreditation network model can be seen in Table 1.

The concept mapping model

In our concept mapping network model, we define four types of entities: Outcome, Concept, Module and Course. In our ontology, a module is a unit of curricular organization smaller than a course. The size of a module may vary, but we find that a typical semester-long course tends to comprise three to six modules. We define three types of relationships: *has-parent-of* relationships, *addresses* relationships, and *leads-to* relationships. The *has-parent-of* relationship is a directed relationship that specifies organizational hierarchy; in this case it relates outcomes to modules, concepts to modules, and modules to courses. The *addresses* relationship is a directed relationship indicating that a concept

addresses an outcome and may be weighted to indicate strength of connection. The *leads-to* relationship is a directed relationship between outcomes and is used to represent prerequisite relationships among outcomes as described in the outcomes mapping framework of [20]. Thus, our concept map represents relationships between outcomes and concepts, as well as relationships among the outcomes themselves. Figure 6 shows this concept mapping model ontology.

Our visualization application for the concept map draws nodes for concepts and outcomes. Using the specified *has-parent-of* relationships, we visualize concepts and outcomes grouped into modules, and modules grouped into their respective courses. We draw a directed edge from a concept node to an outcome node to visualize an *addresses* relationship. We draw a directed edge from one outcome node to another outcome node to show a *leads-to* relationship. The types of entities and relationships for the concept map network model can be seen in Table 1.

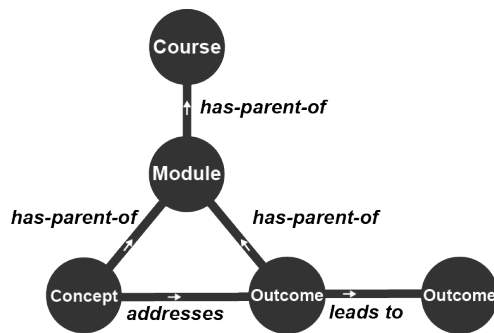


Figure 6: A concept mapping model ontology.

Educational mapping

In this section we define and describe our process of mapping. Here, we define the mapping process as the process of transforming an initial data set into a *mapped* data set consisting of entities and relationships as described in our network models.

Table 2 shows a list of notional raw input data. The input data may come in many forms, but often its form will be similar to that depicted in Table 2. Mapping is the process of converting this data set into a structured form according to the mathematical network models presented in the previous section.

Table 2: Before mapping: input data.

Id	Name	Type	Prerequisites
entity-A	An entity in the data set	Type 1	entity-C, entity-F
entity-B	Another entity	Type 2	entity-C

The first step in mapping is to identify the entities of interest in the system. We step through the data set and construct entity objects. We assign each entity

a unique identifier, and attach to it its type and other attributes. The constructed entity object may be minimally described in plain-text JSON as:

```
1 {
2   "id": "entity-A",
3   "name": "An entity in the data set",
4   "type": "Type 1"
5 }
```

The second step in mapping is to construct relationship objects—we emphasize the construction of relationships as explicit objects. Current modeling formalisms flatten relationships to be attributes of an entity, as seen in Table 2. As discussed in [1], such attribute-based modeling makes it difficult to analyze the ontological structure of the data and requires ad-hoc mechanisms to derive inferences. For example, attribute-based modeling makes it difficult to ask questions about pathways such as, what entities are joined to other entities that are in themselves joined to other entities via prerequisites? The explicit modeling of relationships is a key contribution of our educational modeling work. We create an object for every relationship that appears, and assign the relationship a unique identifier. We also assign attributes representing the directionality, weighting, cost and other attributes of each relationship. A relationship object may be minimally represented in JSON as:

```
1 {
2   "id": "edge-has-prerequisite-of-AF",
3   "type": "has-prerequisite-of",
4   "genus": "DIRECTED",
5   "sourceId": "entity-A"
6   "targetId": "entity-F",
7   "weight": 1
8 }
```

In this JSON example, the fields `sourceId` and `targetId` point to the unique identifiers of the source and target entity objects. Additional attributes may be present, depending on the data and application use case.

Results

In this section we present three case studies to demonstrate application of our network-based educational models at multiple scales: 1) curriculum mapping at the institutional level, 2) outcomes-based accreditation at the degree program level and 3) instructional planning at the course level. In each case, we describe the mapping process and the resulting network model. We present example visualizations and analytics to illustrate the power of the modelling approach. Our implementation uses a scalable decoupled architecture that enables data to be accessed by multiple independent applications (in our case analytics and visualization) as illustrated in Figure 7.

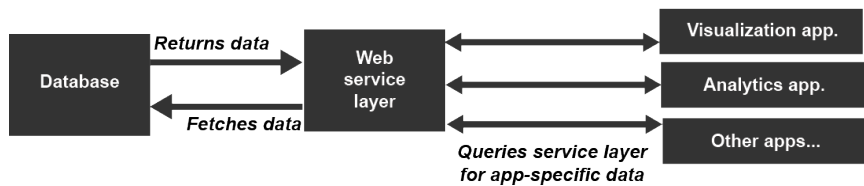


Figure 7: A decoupled architecture consisting of three layers, from left to right: the backend, the web service, and the frontend applications.

Modeling at the institutional scale

In this example, we model the undergraduate curriculum of the Massachusetts Institute of Technology (MIT).

Mapping

The MIT curriculum model uses the structure specified in Table 1. From a curriculum file provided by the MIT registrar office, we create the mapped data set consisting of entities and relationships as described by our network model. This data set represents a snapshot of the MIT undergraduate curriculum as of Fall 2015. Table 3 shows a sample of the input data file to the mapping process.

The first step in the mapping process is to identify the entities. To create the curriculum mapping, we step through each row of the input file and construct an entity of type *Course* for each row. We attach a *units* attribute for each course entity, which indicates the number of units (credits) associated with the course, and a *URL* attribute, which points to the course listing at the official MIT registrar website. We further construct entities of type *Department* for each unique department by parsing the number of the course³. Finally we construct one entity of type *Institution*, representing the institution MIT.

The second step in the mapping process is to construct relationship objects. To do this, we make a second pass through the data and construct a relationship object for every prerequisite and corerequisite requirement that appears in the entity's *Prereqs* and *Coreqs* columns. We also construct a *has-parent-of* relationship object for every *Course–Department* relationship and every *Department–Institution* relationship. Table 4 summarizes the mapped data set.

Analysis

Figure 8 shows a snapshot of the resulting network visualization, in which courses are visualized as nodes, clustered by their parent departments. The zoomed-in views illustrate the prerequisite relationships visualized as directed edges between the appropriate course nodes. In the example shown, we use node color to visualize how courses vary in unit count across the institute. A standard MIT course is 12 units (representing 12 total hours per week over a semester of length 14 calendar weeks). In Figure 8, orange nodes represent courses that are greater than 12 units—these are typically laboratory and project-based courses. As well as laboratory and project-based courses across the engineering and science departments, we see in Global Studies and Languages several courses that include research projects conducted in the relevant foreign

³At MIT, courses begin with the number of their department. For example, the course 2.007 belongs in Course 2 Mechanical Engineering.

Table 3: Before mapping: sample of MIT curriculum file.

Name	Prereqs	Coreqs	Units
2.007 Design and Manufacturing I	2.001,2.670	2.086	12
2.670 Mechanical Engineering Tools			3
6.0001 Intro. to Comp. Sci. Prog. in Python			6
6.005 Elements of Software Construction	6.01	6.042	12
6.009 Fundamentals of Programming	6.0001		12
15.301 Managerial Psychology Laboratory			15
...

Table 4: After mapping: summary of mapped MIT curriculum data set.

Entities	Count	Attributes
Course	1264	<i>Units, URL</i>
Department	31	
Institution	1	
Relationships	Count	Attributes
has-parent-of	1295	<i>Type, Directionality</i>
has-prerequisite-of	941	<i>Type, Directionality</i>
has-corequisite-of	63	<i>Type, Directionality</i>

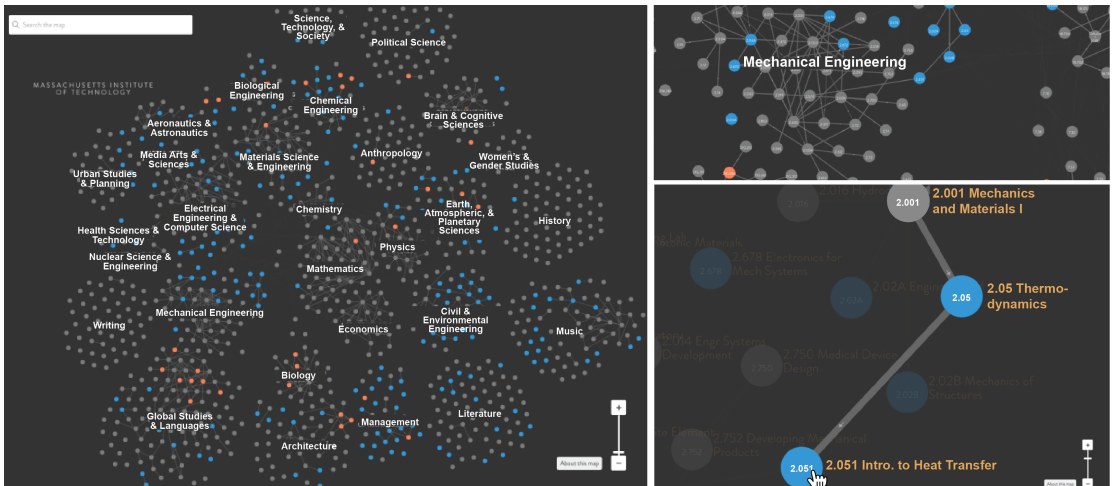


Figure 8: Visualization of MIT curriculum mapping: grey nodes indicate courses with 12 units (a standard semester-long course), blue nodes indicate courses with fewer than 12 units, and orange nodes indicate courses with more than 12 units. Here, three zoom levels of the visualization are shown.

language. Blue nodes represent courses that are fewer than 12 units. Here, we see evidence of the recent curricular redesigns of several MIT departments to include more flexibility in their undergraduate degree programs. For example, Mechanical Engineering at MIT offers a both a traditional and a flexible degree program. In the flexible degree program, students complete a core in mechanical engineering and combine it with a six-course concentration in one of several modern engineering areas. In part, this flexibility is enabled through half-semester courses (6 units) in the mechanical engineering core. The highlighted pathway in Figure 8 shows a full-semester course Mechanics and Materials I leading to a half-semester course Thermodynamics, which in turn leads to another half-semester course Introduction to Heat Transfer. Mechanics and Materials I is required for students in both the traditional and the flexible mechanical engineering degree programs. The two follow-on six-unit courses are required for the flexible degree program (whereas the traditional degree program has a different requirement), but the offering as two half-semester courses is intended to give the students greater scheduling flexibility to accommodate their broader degree requirements.

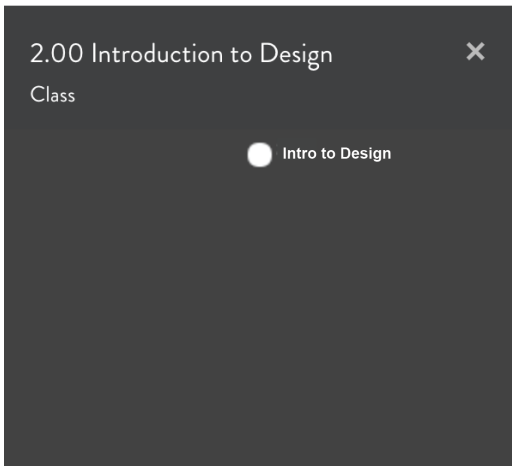
Another analysis of the curriculum model is of prerequisite relationships. For each course, we find its entire prerequisite chain—i.e., the course's prerequisites, the prerequisites of its prerequisites, and so on. This prerequisite chain is a subgraph within our network model. We then compute a topological sort on the subgraph to find a valid ordering of the prerequisite pathway of the course. Figure 9 visualizes some prerequisite pathways in a tree-like structure. In the visualization, nodes are ranked according to the maximum length of the pathway from the source node(s) in the subgraph to that node. Shown is a collection of prerequisite pathways with courses ordered by increasing rank. Note that in the interest of presenting a simple illustrative graphic, these visualizations do not attempt to represent “or” conditions in the prerequisite pathways.

Modeling at the program scale

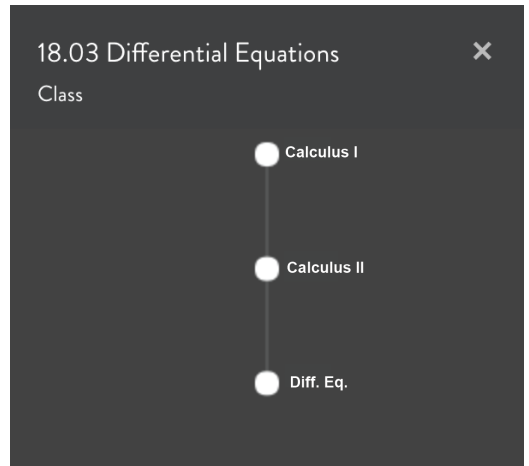
In this example, we model an individual program—the Engineering Product Development (EPD) degree program at the Singapore University of Technology and Design (SUTD). We discuss how our network model can be used to support accreditation analysis.

Mapping

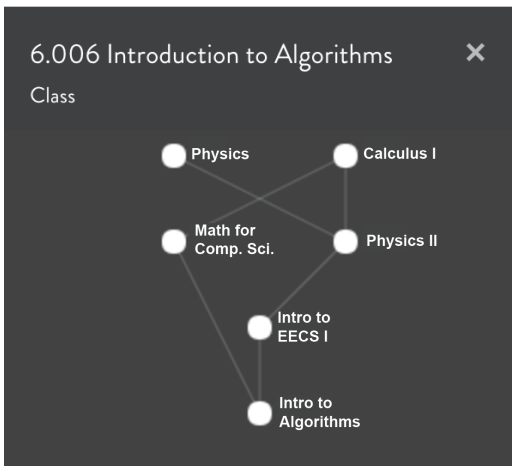
The SUTD EPD program model uses the structure specified in Table 1. We obtained from SUTD a list of program learning outcomes and courses, organized into groups according to the type of course. Each course has a list of outcomes that it addresses. The degree to which a course addresses an outcome is indicated by a numeric weighting from one (addresses weakly) to three (addresses strongly). To create the accreditation mapping network model, we step through the list to construct entity objects for each outcome, course, grouping and degree program that appear. These constructed entities are vertices in the network model and have type of Outcome, Course, Group or Program. We also attach attributes to each entity, such as an URL for each course that directs the user to a repository of materials for that course, including evidence that is provided to accreditors during a site visit.



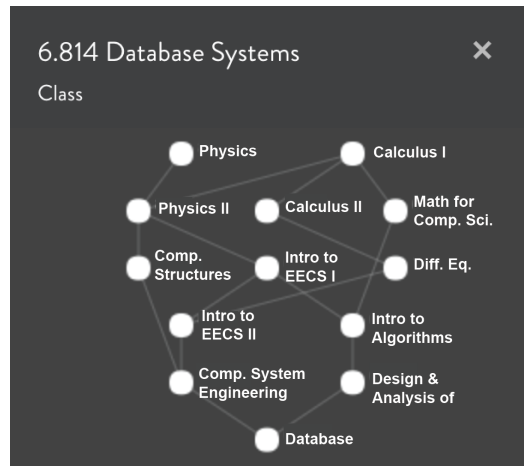
(a) The course 2.00 Introduction to Design has a rank of 0.



(b) The course 18.03 Differential Equations has a rank of 2.



(c) The course 6.006 Introduction to Algorithms has a rank of 3.



(d) The course 6.814 Database Systems has a rank of 5.

Figure 9: These panels visualize the prerequisite pathways of courses. Within a pathway, course nodes are ordered by increasing rank.

The second step in the mapping process is to construct relationship objects. To do this, we make a second pass through the list and construct a relationship object of type *addresses* pointing from each course to the corresponding outcomes. These relationships are each assigned a weighting. We construct a *has-parent-of* relationship object for every Course–Group, Outcome–Group, and Group–Program relationship.

Table 5 summarizes the mapped data set and Figure 10 shows a snapshot of the resulting network visualization. In our visualization application, we visualize outcomes as nodes (small red circles) and courses as nodes (larger circles). The color of a course node indicates the number of outcomes it addresses, with brighter nodes addressing more outcomes. Groups are used to visually cluster

outcomes and courses using *has-parent-of* relationships and for suburb labels in the map. The *addresses* relationships are shown as arrows pointing from a course node to an outcome node.

Table 5: After mapping: summary of the mapped SUTD EPD data set.

Entities	Count	Attributes
Outcome	40	<i>Description</i>
Course	85	<i>Learning evidence URL</i>
Group	7	
Program	1	
Relationships	Count	Attributes
has-parent-of	132	<i>Type, Directionality</i>
addresses	1559	<i>Type, Directionality, Strength</i>

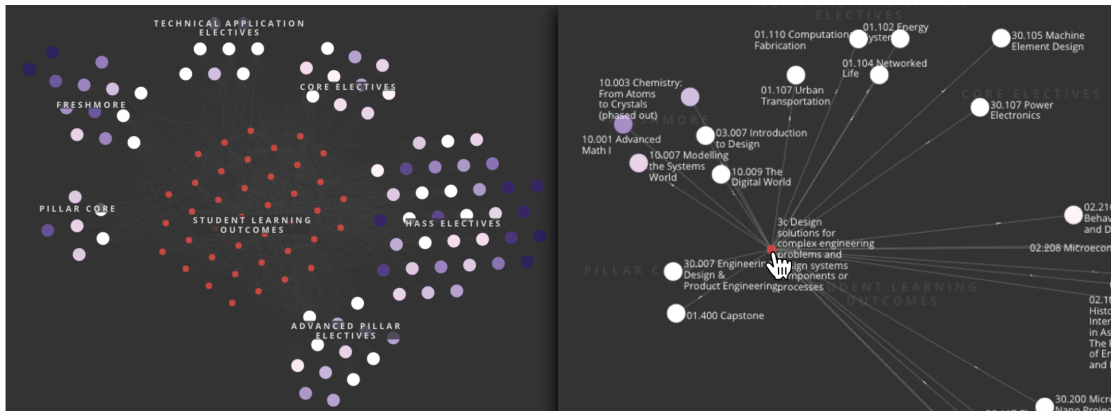


Figure 10: Visualization of SUTD Engineering Product Development degree program mapping.

Analysis

This network model provides a basis on which to conduct analysis of the program and its coverage of accreditation outcomes. For each outcome we can analyze its coverage in the EPD program. The indegree of an outcome vertex specifies the number of courses that address that outcome. The weighted indegree of an outcome vertex is computed by summing up the weights of the incoming edges. This gives an indication of the strength of coverage of that outcome across the curriculum. For example, the outcome “*Identify responsibilities relevant to professional engineering practice through a clear needs statement in capstone or design projects*” is addressed by the following courses (weighting shown in parentheses): Introduction to Design (1), Modelling the Systems World (1), The Digital World (1), Engineering Design & Project Engineering (2), Capstone (3),

Entrepreneurship (3), Power Electronics (1), Design & Fabrication of MEMS (1), Electric Power Systems Analysis and Design (1), Micro-Nano Projects Laboratory (2), Digital Integrated Circuits Design (1), Topics in Biomedical & Healthcare Engineering (1), Design Management (3), Design and Manufacturing (1), Engineering Management (2), Culture Formation and Innovative Design (3), Energy Systems (3), Urban Transportation (1), The History of International Development in Asia: The Role of Engineers and Designers (1), Social Theories of Urban Life (1), Who Gets Ahead? Sociology of Social Networks and Social Capital (2), Rice Cultures: Technology, Society, and Environment in Asia (2), How the Things People Make, Make People: Material Things in Social Life (1). This gives a weighted indegree score for the outcome vertex of 38.

Figure 11 plots the weighted indegree scores for all 40 outcomes. These data emphasize the strongly interdisciplinary and design-focused nature of the SUTD curriculum, with a strong mapping between courses and outcomes. This strong mapping is not coincidental—SUTD is a newly founded university and its curriculum was designed from scratch to emphasize cross-cutting skills and challenges, rather than the traditional disciplinary approach of most engineering programs. In general, this kind of network analysis will reveal potential gaps of coverage in a curriculum, although in the SUTD EPD case, no such gaps are apparent.

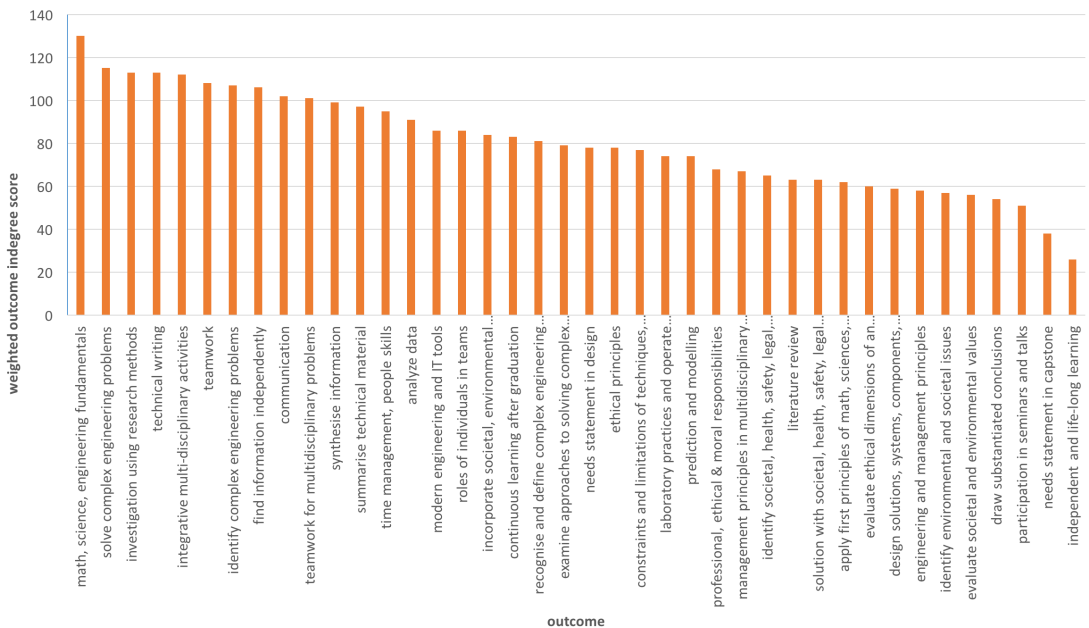


Figure 11: The weighted indegree score for each outcome shows how strongly the outcome is addressed by courses across the SUTD EPD curriculum.

An analysis of course vertices reveals how strongly a particular course contributes to addressing program outcomes. Similar to the weighted indegree calculation described above, we can compute the weighted outdegree of a course vertex by summing the weights associated with all outgoing *addresses* relationship edges. Figure 12 plots the results for the 20 courses in the

SUTD EPD curriculum that have the highest outdegree scores. These analytics are useful, for example, in considering the effects of curriculum redesign on coverage of program outcomes. These scores could also be combined with student enrolment data to provide a quantitative assessment of student coverage of outcomes, either by individual student or in aggregate across the student population.

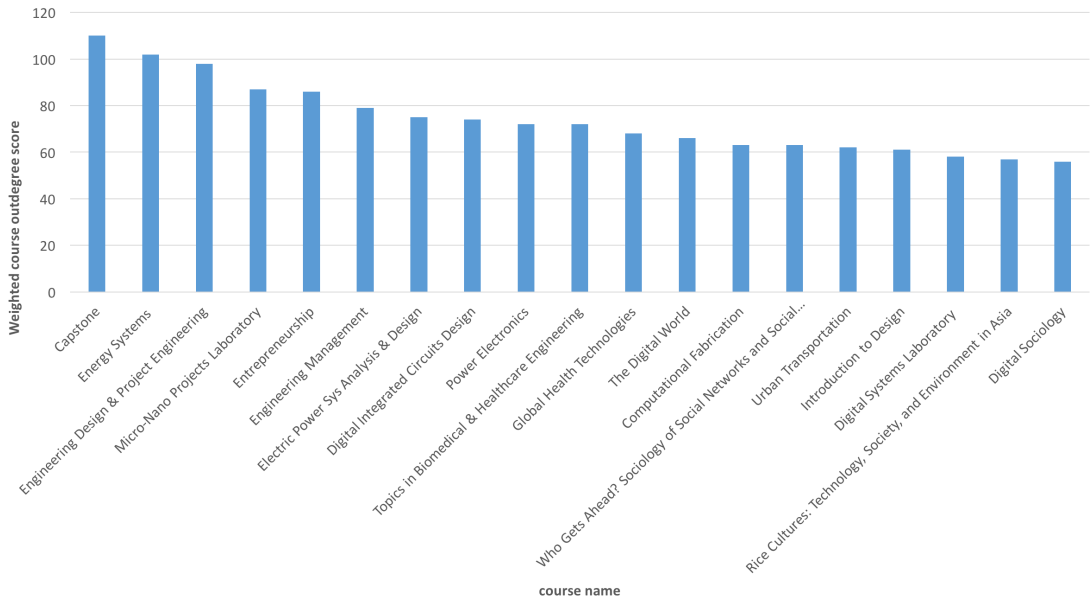


Figure 12: The weighted outdegree score for each course shows how strongly the course contributes to outcomes across the SUTD EPD curriculum.

Modeling at the course scale

In this example we model and map a single course, Computational Methods in Aerospace Engineering.

Mapping

The concept map model uses the structure specified in Table 1. The course has 59 learning outcomes describing what a student is expected to be able to do after completing the course. The instructor identified 67 concepts, highlighting what she thought to be the key topics covered in the course. These outcomes and concepts are grouped into four modules: Integration Methods for Ordinary Differential Equations (ODEs), Finite Difference and Finite Volume Methods for Partial Differential Equations (PDEs), Finite Element Methods for PDEs, and Probabilistic Simulation and Intro to Design Optimization. Within each of these modules, the concepts and outcomes are each grouped into a sub-module. This leads to the concept map network model with a total of 139 entities as shown in Table 6.

The second step in the mapping process is to construct relationship objects. Relationships contained in the provided course data include the relationships

between concepts and outcomes; these specify which outcomes each concept addresses. The data also contain information on the relationships among outcomes, by specifying for each outcome a list of its prerequisite outcomes (i.e., the other outcomes that must be mastered in order to achieve that particular outcome).

To construct the relationship objects in the network model, we step through each concept and identify the outcomes it addresses. For each, we construct a relationship object of type *addresses* pointing from the concept to the outcome. We step through each outcome and construct a relationship object of type *leads-to* pointing from the outcome to any downstream outcomes for which it is a prerequisite. We construct a *has-parent-of* relationship object for every Concept–Module, Outcome–Module, and Module–Course grouping. Table 6 summarizes the mapped dataset and Figure 13 shows a snapshot of the resulting network visualization.

Table 6: After mapping: summary of mapped dataset for course Computational Methods in Aerospace Engineering.

Entities	Count	Attributes
Outcome	59	
Concept	67	
Module	12	
Course	1	
Relationships	Count	Attributes
has-parent-of	138	<i>Type, Directionality</i>
addresses	157	<i>Type, Directionality</i>
leads-to	50	<i>Type, Directionality</i>

Analysis

As with the accreditation map example above, an analysis of the indegree of outcomes and outdegree of concepts reveals how course content covers the course learning outcomes. One could further model the relationship between course assessments (exams, projects, homeworks, etc.) and outcomes. Layering student assessment results could then provide granular insight into student achievement of course learning outcomes.

As one example of analysis on the concept mapping network model, we consider the subgraph corresponding to only the *leads-to* relationships and perform a topological sort of the subgraph. We then rank the outcomes according to the length of their prerequisite chains. The outcomes with the highest rankings are those that have the most prerequisites, and thus represent skills that build upon and synthesize upstream skills. In Computational Methods in Aerospace Engineering, the highest rank of an outcome is four (i.e., there is at least one path that includes four prerequisite outcomes that *lead to* this outcome). The three outcomes of rank four are shown in Table 7. The table also shows the total number of prerequisite outcomes for each outcome (including

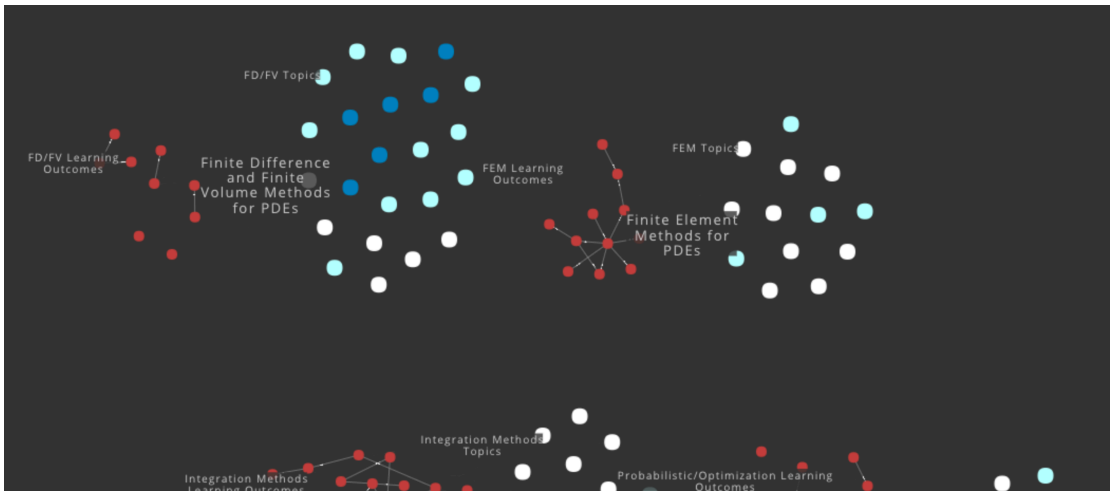


Figure 13: Visualization of concept map for course Computational Methods for Aerospace Engineering.

inherited prerequisites, i.e., prerequisites of prerequisites). For example, the outcome “Implement multi-step and multi-stage methods to solve a representative system of ODEs from an engineering application” is a synthesizing outcome that requires students to have mastered upstream skills such as determining a method’s convergence properties, determining stability boundary, and explaining the difference between explicit and implicit methods. Figure 14 shows a tree visualization of this outcome and its prerequisite outcomes within the course. The rank of four is apparent from the number of levels in the tree-like structure.

Table 7: Outcomes of high rank are synthesizing skills that build on earlier material in the course Computational Methods in Aerospace Engineering.

Learning Outcome	Rank	Total number of prerequisite outcomes
Implement multi-step and multi-stage methods to solve a representative system of ODEs from an engineering application.	4	13
Obtain confidence intervals for sample estimates of the mean, variance, and event probability.	4	9
Describe the meaning of the entries (rows and columns) of the stiffness matrix and of the right-hand side vector for linear problems.	4	9

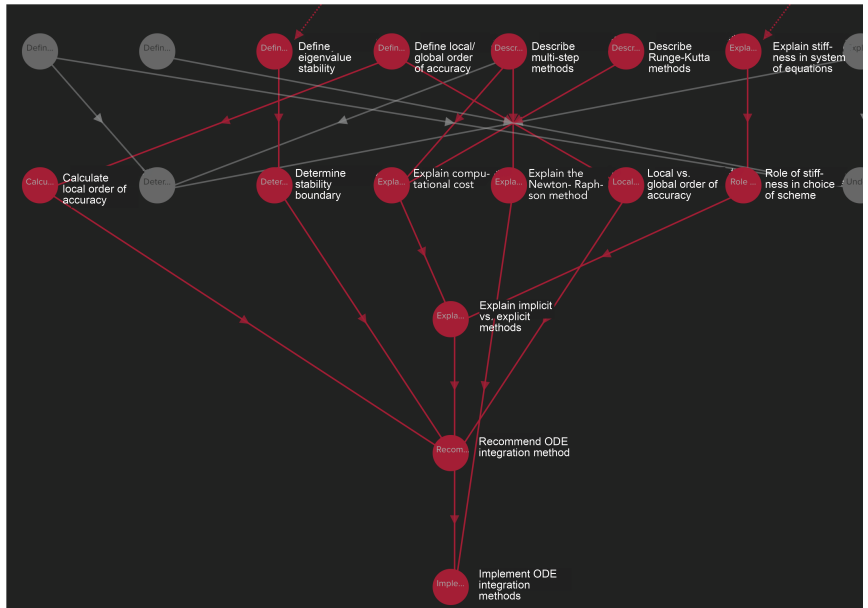


Figure 14: Visualization of the tree-like structure yielded by the outcome *Implement ODE integration methods* and its prerequisite chain.

Conclusion

This paper has presented a flexible and scalable mathematical framework for educational modeling, with modeling examples ranging from an institutional-wide curriculum to an engineering degree program to a single course. The resulting network models embody the mappings that represent the rich relationships among different educational entities, such as courses, concepts, outcomes, departments and degree programs. The network models enable visualization and analytics, either using standard graph visualization and graph analytics tools, or using components tailored to the specific educational setting. In the examples presented, viewing the elements of an educational curriculum through the structured lens of a network model provides insight into learning pathways and permits gap analysis of curriculum coverage, supporting curricular design.

For privacy reasons, this paper has avoided any examples that use actual student data; however, clearly the presented models provide a structured foundation that, in concert with student data, could enable adaptive learning, personalized learning and data-driven institutional resource allocation.

Acknowledgements

The authors thank Ri Romano of MIT and Toe Oo Zaw of SUTD for their help in collecting the data used in the examples.

References

- [1] A. Artale, E. Franconi, N. Guarino, and L. Pazzi. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, 20(3):347–383, November 1999.

- [2] T. Bailey, S. Jaggars, and D. Jenkins. *Redesigning America's Community Colleges: A Clearer Path to Student Success*. Harvard University Press, Cambridge, Massachusetts, 2015.
- [3] A.-L. Barabasi and Z. Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.
- [4] M. Bell and Y. Lida. *Transportation Network Analysis*. Wiley (John) & Sons, Sussex, England, 1997.
- [5] B. Bollobás. *Modern Graph Theory*. Springer Science & Business Media, New York, 1998.
- [6] J. Bondy and U. Murty. *Graph Theory with Applications*. Macmillan, London, 1976.
- [7] J. Campbell and D. Oblinger. Academic analytics. *EDUCAUSE Review*, 42(4):40–57, 2007.
- [8] P. Carrington, J. Scott, and S. Wasserman. *Models and Methods in Social Network Analysis*. Cambridge University Press, New York, 2005.
- [9] A. Carstensen and J. Bernhard. Design science research as an approach for engineering education research. *Proceedings of the 12th International CDIO Conference, Turku University of Applied Sciences, Turku, Finland, June 12-16, 2016*, June 12-16 2016.
- [10] A. Clauset, A. Arbesman, and D. Larremore. Systematic inequality and hierarchy in faculty hiring networks. *Science Advances*, 1(1), February 2015.
- [11] A. Degenne and M. Forsé. *Introducing Social Networks*. Sage Publications, London, 1999.
- [12] N. Guarino. Concepts, attributes and arbitrary relations: some linguistic and ontological criteria for structuring knowledge bases. *Data & Knowledge Engineering*, 8(3):249–261, July 1992.
- [13] D. Jenkins and S.-W. Cho. Get with the program ... and finish it: Building guided pathways to accelerate student completion. *New Directions for Community Colleges*, 2013(164):27–35, 2013.
- [14] K. Kelley, J. McAuley, L. Wallace, and S. Frank. Curricular mapping: Process and product. *American Journal of Pharmaceutical Education*, 72(5), September 2008.
- [15] D. Lazer, A. Pentland, L. Adamic, S. Aral, A. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. Life in the network: the coming age of computational social science. *Science*, 323(5915):721–723, 2009.
- [16] J. Novak and D. Gowin. *Learning How to Learn*. Cambridge University Press, 1984.
- [17] C. Plaza, J. Draugalis, M. Slack, G. Skrepnek, and K. Sauer. Curriculum mapping in program assessment and evaluation. *American Journal of Pharmaceutical Education*, 71(2), 2007.
- [18] A. Pujol, R. Mosca, J. Farrés, and P. Aloy. Unveiling the role of network and systems biology in drug discovery. *Trends in Pharmacological Sciences*, 31(3):115–123, 2010.
- [19] M. Scott. *Programming Language Pragmatics*. Morgan Kaufmann Publishers, San Francisco, CA, 2006.
- [20] J. Seering, K. Willcox, and L. Huang. Mapping outcomes in an undergraduate aerospace engineering program. In *Proceedings of the American Society for Engineering Education 12th Annual Conference & Exposition, Seattle, WA, June 2015*.

- [21] G. Siemens and P. Long. Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5):30–40, 2011.
- [22] J. Turns, C. J. Atman, and R. Adams. Concept maps for engineering education: A cognitively motivated tool supporting varied assessment functions. *IEEE Transactions on Education*, 43(2):164–173, May 2000.
- [23] D. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, 2001.