Network models for mapping educational data

Karen E. Willcox,[*] Luwen Huang[†]
Massachusetts Institute of Technology
Cambridge, MA 02139

May 2017

Abstract

Educational mapping is the process of analyzing an educational system to identify entities, relationships and attributes. This paper proposes a network modeling approach to educational mapping. Current mapping processes in education typically represent data in forms that do not support scalable learning analytics. For example, a curriculum map is usually a table, where relationships among curricular elements are represented implicitly in the rows of the table. The proposed network modeling approach overcomes this limitation through explicit modeling of these relationships in a graph structure, which in turn unlocks the ability to perform scalable analyses on the dataset. 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 student advising, student degree planning and curricular design at scales ranging from an entire institution to an individual course.

1 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 [14, 2, 18], 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 [25, 13]. In evidence-based frameworks, studying linkages within and across learning processes are critical to informing instructional methods and changes in student knowledge [16]. By analyzing the linkages within connected data, we can move towards learning engineering [21] and better design educational experiences [9, 18].

[*]Professor, Department of Aeronautics and Astronautics, kwillcox@mit.edu
[†]The Mapping Lab, luwenh@mit.edu

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]. For example, a curriculum map is typically represented as a table, with topics or courses (the "entities") listed in the rows of the table. Related data such as assessments, standards addressed, instructor, etc. are listed in the columns of the table. In this tabular representation, each row is recognized as an entity and given a first-class representation. However, an entity's relationships are implicitly defined—flattened into column attributes of that row and thus not specified as first-class objects in their own right. This way of representing information means that analysis algorithms must be written in case-specific manner, requiring a potentially different analysis code for each new dataset. 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, 17], biological networks [3, 20], and 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. In the curriculum map example described above, a network model would explicitly represent as entities the topics, courses, assessments, standards, instructors, etc., and it would also explicitly represent the various relationships among these different kinds of entities. The reader could imagine multiple tables or spreadsheets listing all of these entities and all these relationships—while it might appear that the network model is a less compact representation than the traditional table, in fact this expanded representation is far more flexible and scalable. In this paper, 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, 24]. 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.

2 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.

[1]For an introductory background on the notion of first-class citizenship in computer science, we refer the reader to [22].

2.1 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. For example, in a curriculum map the entities might be topics, courses, assessments, standards, schedule information, and/or students. Relationship edges might represent information such as which topics are covered in each course, which courses a particular student has completed, which assessments relate to which topics, and so on. Explicit modeling of these relationships in the graph structure unlocks the ability to perform scalable analyses on the dataset, such as a pathway analysis of topics to which each student has been exposed and the students associated level of competency for each topic. Such an analysis can be invaluable in student advising and career planning. Yet it is infeasible in current mapping processes where tabular models flatten relationships into implicit representations.

In the graph structure, 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. For example, Course B may require Course A as a prerequisite. This would be represented as a directed edge pointing from Course A to Course B. An *undirected edge* is bidirectional—the relationship applies in both directions. For example, a topic may be related to another topic without any sense of ordering. Visually, this undirected edge 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.

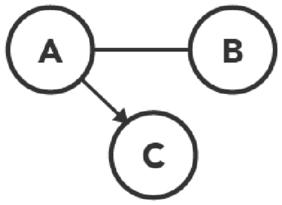


Figure 1: A graph structure G with three vertices and two edges. The edge from vertex A to vertex C is directed, while the edge between vertex A and vertex B is undirected.

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, 26] for more detail.[2]

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. These notions will be useful, for example, when assessing how the content in

[2]Wikipedia also has a useful "Glossary of graph theory terms" `https://en.wikipedia.org/wiki/Glossary_of_graph_theory_terms`.

an educational program relates to student learning outcomes, or when assessing how a particular student's coursework addresses accreditation outcomes. These concepts will be illustrated further in the accreditation mapping and concept mapping examples presented later in this paper.

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. This analysis is useful, for example, when considering the prerequisite relationships between courses. In that case, the maximum length of the pathway associated with a particular course represents the most constraining set of prerequisite requirements that the student must complete before they can take that course. This provides valuable information for student advising and degree planning. This will be illustrated further in the curriculum mapping examples presented later in this paper.

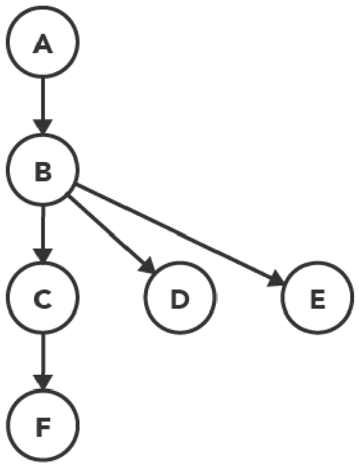


Figure 2: A directed graph with vertices arranged by assigned rank. Relative to the source vertex (Vertex A): Vertex A has a rank of 0; Vertex B has a rank of 1; Vertex C, Vertex D and Vertex E have a rank of 2; and Vertex F has a rank of 3.

2.2 The curriculum mapping model

We define a curriculum mapping network model as follows. First, we define the different types of entities that reflect organization of the curriculum. As a first simple model to illustrate the ideas, consider a curriculum of a single institution, comprising a large number of courses arranged into departments. In this example, there are three different types of entities: a Course, a Department, and an Institution. Categorizing the entities by different types will prove useful when we come to define relationships. Second, we define all the individual entities in the curriculum model: in this example, we define the individual courses, each of the departments, and the institution. Each one of these entities is modelled as a vertex in the curriculum network model. Third, each entity is assigned attributes. For example, a Course entity might have attributes that includes its name, course number, schedule information, website listing, etc. A Department entity might have attributes that includes its name, departmental code, website listing, department head, etc.

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[3] Again, these relationships are modelled as directed edges between the appropriate vertices in the curriculum network model.

To give a sense of scale of the model, if the curriculum comprises n courses across m departments and a single institution, then the total number of vertices in the network model is $n+m+1$. 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. For example, if an institution's curriculum comprises 1000 courses across 25 departments with 1200 prerequisite relationships and 50 corequisite relationships, then our curriculum mapping network model has 1026 vertices and 2275 directed edges.

Figure 3 shows the curriculum model ontology corresponding to the network model described above. The left side of the figure shows the general ontology and the right side of the figure shows a concrete example. In this particular example ontology, a course is the most granular entity, 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. One of the advantages of the network modeling approach is its flexibility to accommodate as many different types of entities and relationships as the modeler wishes to define for the particular use case.

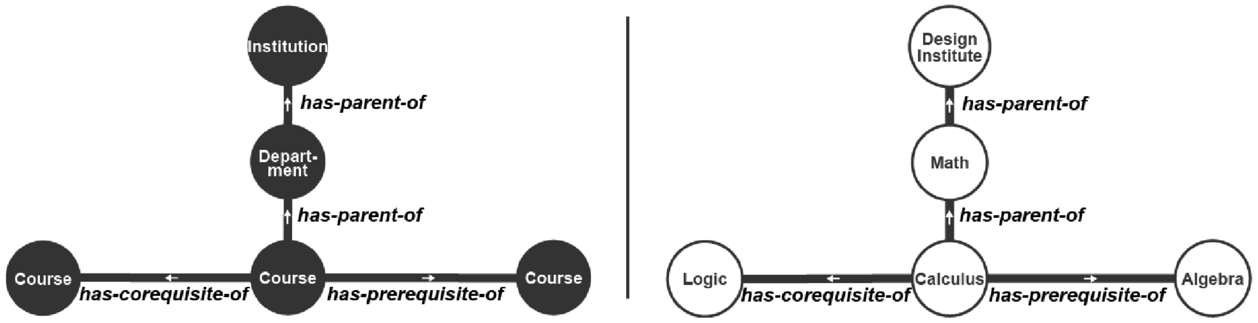


Figure 3: A curriculum mapping model ontology. Left: the general ontology. Right: a concrete example.

This modeling approach can be used to model other, more complicated aspects of a curriculum. For example, sometimes multiple courses can fulfill a given prerequisite requirement. In this case,

[3]A prerequisite is another course that must be taken before enrollment in the course at hand. A corequisite is a course that must be taken concurrently with (or prior to) enrollment in the course at hand.

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 $\mathcal{C} = \{c_1, c_2, \ldots c_s\}$, 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. $\mathcal{C}$ can then be modelled by a single vertex labeled `OR`. We attach a *required-number* attribute to the vertex with value ℓ. This *required-number* attribute specifies that a student must complete ℓ prerequisite courses from the specified set. 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 $\mathcal{C}$, via *can-be* relationships. This modeling is illustrated in Figure 4.

Note also that this model extends easily to the case of multiple "or" requirements. For example, consider the case that an organic chemistry course (Org. Chem) has a prerequisite requirement that a student has completed one calculus course, chosen from a defined set of three alternative calculus classes (Calc_A, Calc_B or Calc_C), and one chemistry course, chosen from a defined set of three alternative chemistry classes (Chem_A, Chem_B or Chem_C). To model this situation, we would define two `OR` vertices, each attached to the Org. Chem course vertex via a *has-prerequisite-of* relationship. One `OR` vertex would be connected via *can-be* relationships to each of the three calculus classes and would have a *required-number* attribute of one. The other `OR` vertex would be connected via *can-be* relationships to each of the three chemistry classes and would have a *required-number* attribute of one. This example is illustrated in Figure 4.

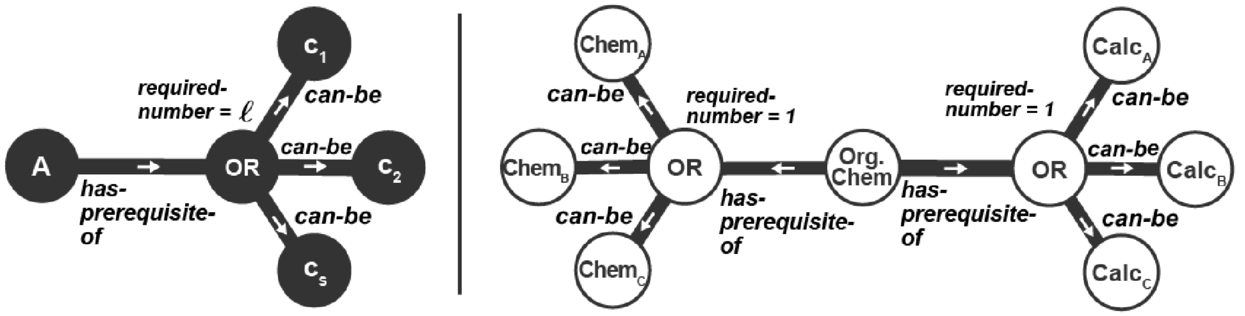


Figure 4: Capturing the situation of alternate prerequisite requirements by introducing an `OR` vertex and *can-be* relationship to the network model. Left: the general ontology. Right: a concrete example.

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, that is, 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. For example (to be expanded on in the results), one can discover courses that have long constraining prerequisite chains and are thus inaccessible to a large portion of students. These kinds of analyses could have useful applications in student advising and degree planning, as well as in curriculum planning and reform.

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, because this visual arrangement reflects the organizational structure of the institution and it also reflects the familiar structure of online course listings. 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	*has-parent-of* *has-prerequisite-of* *has-corequisite-of* *can-be*	*has-prerequisite-of* *has-corequisite-of*
Accreditation mapping	Outcome Course Group Program	Outcome Course	*has-parent-of* *addresses*	*addresses*
Concept mapping	Outcome Concept Module Course	Concept Outcome	*has-parent-of* *leads-to* *addresses*	*leads-to* *addresses*

2.3 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 [15, 19].

We define an accreditation mapping network model as follows. We define four different types of entities: Outcome, Course, Group and Program. An “outcome” here refers to an outcome used as a criterion in an accreditation study, such as the Accreditation Board for Engineering and Technology (ABET) student outcomes. These outcomes are typically defined by an external accreditation agency. A “group” here refers to a way in which courses might be grouped (for example, as “required courses,” or “elective courses,” or “capstone courses,” etc.). A “program” refers to a program of study, typically a degree program. We then define all the individual entities in the accreditation mapping network model: we define the individual outcomes, the individual courses, the groups, and the program. Each one of these entities 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.

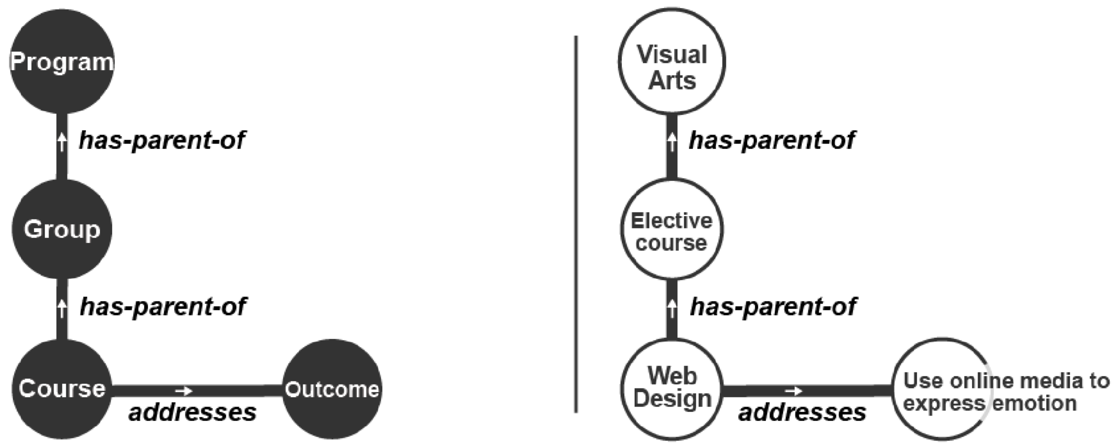


Figure 5: The accreditation mapping model ontology. Left: the general ontology. Right: a concrete example.

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.

2.4 The concept mapping model

In our concept mapping network model, we define four different 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. An "outcome" here may refer to an accreditation outcome as in the accreditation mapping network model, but more often it refers to a learning outcome. Such learning outcomes are sometimes prescribed by an educational institution or governing body (for example, as is the case in some community college systems), or they may be authored by individual course instructors. We use the term "concept" to denote the main idea underlying a (typically small) unit of content covered in the course. In our terminology, a concept is one example of a so-called "Knowledge Component," as described in the Knowledge-Learning-Instruction framework in [16].

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 [23]. 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.

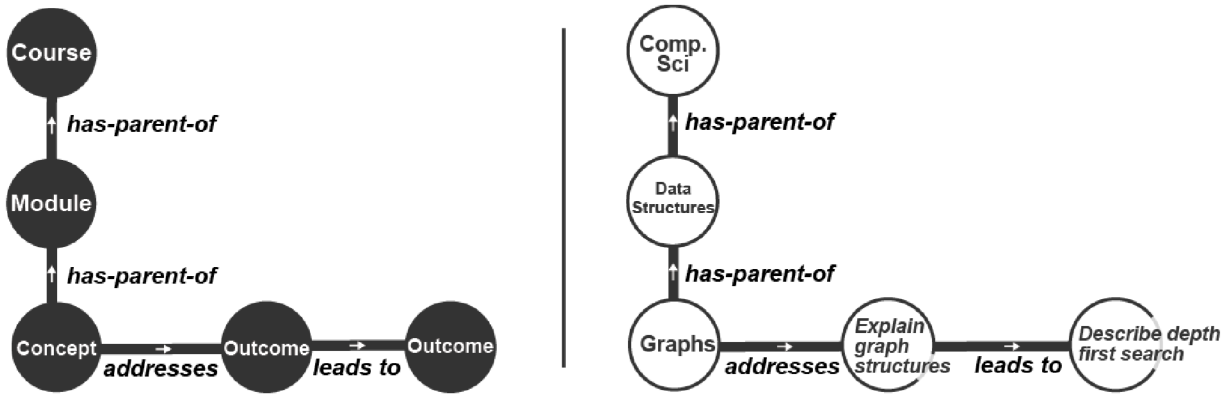


Figure 6: A concept mapping model ontology. Left: the general ontology. Right: a concrete example.

3 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	Differential Equations	Course	
entity-B	Dynamics	Course	entity-A
entity-C	Signals and Systems	Course	entity-A
entity-D	Feedback Control	Course	entity-C
entity-E	Robotic Design	Course	entity-B, entity-D

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 JavaScript Object Notation (JSON)[4] as:

```
{
  "id": "entity-A",
  "name": "Entity A",
  "type": "Course"
}
```

To illustrate with a concrete example, a JSON representation of the entity in the third row of Table 2 can be:

```
{
  "id": "entity-C",
  "name": "Signals and Systems",
  "type": "Course"
}
```

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

[4]JSON is a platform-independent way of representing data, easy for humans to read and write. Sample open-source data sets in JSON and Excel are available for the examples in this paper at mapping.mit.edu/network-models-for-education

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 courses are joined to other courses that are in themselves joined to other courses via prerequisites? The explicit modeling of relationships is a key contribution of our educational modeling work. Our graph model explicitly models relationships and entities, enabling reusable analyses at large scale.

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. For example, a relationship object such as the prerequisite requirement specified in the third row of Table 2 may be minimally represented in JSON as:

```json
{
  "id": "edge-has-prerequisite-of-CA",
  "type": "has-prerequisite-of",
  "genus": "DIRECTED",
  "sourceId": "entity-C"
  "targetId": "entity-A",
  "weight": 1
}
```

In this JSON example, the fields `sourceId` and `targetId` point to the unique identifiers of the source and target entity objects (here the Signals and Systems course with ID entity-C has a prerequisite of the Differential Equations course with ID entity-A). The field `genus` indicates whether the relationship is directed or undirected. Additional attributes may be present, depending on the data and application use case.

4 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.

4.1 Modeling at the institutional scale

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

4.1.1 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

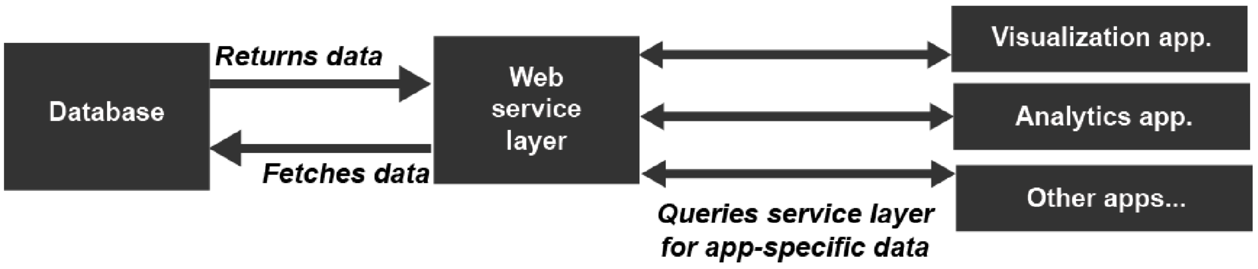


Figure 7: A decoupled architecture consisting of three layers, from left to right: the backend, the web service, and the frontend applications.

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 course[5]. 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.

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
...	...	...	...

4.1.2 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

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

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

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

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 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 the maximum length of the pathway represents the most constraining set of prerequisite requirements that the student must complete before they can take that course. For example, in Figure 9(c), the course Intro to EECS I is shown in the third level of the tree-like structure, because students must complete two levels of prerequisite classes (here Physics II and its prerequisites) before taking Intro to EECS I. Similarly, the course Intro to Algorithms has a

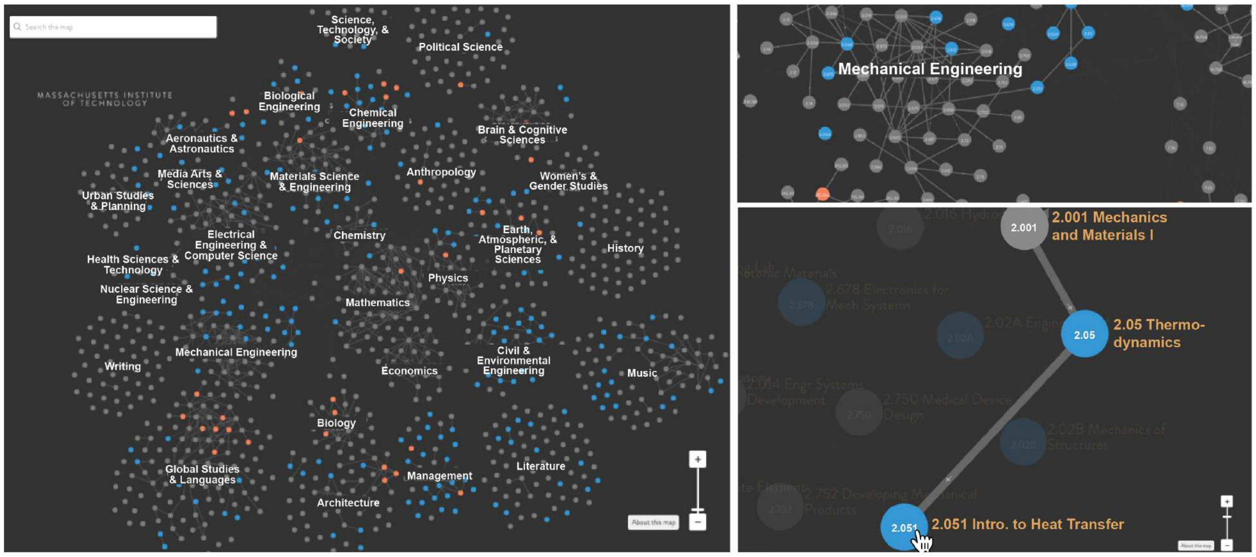
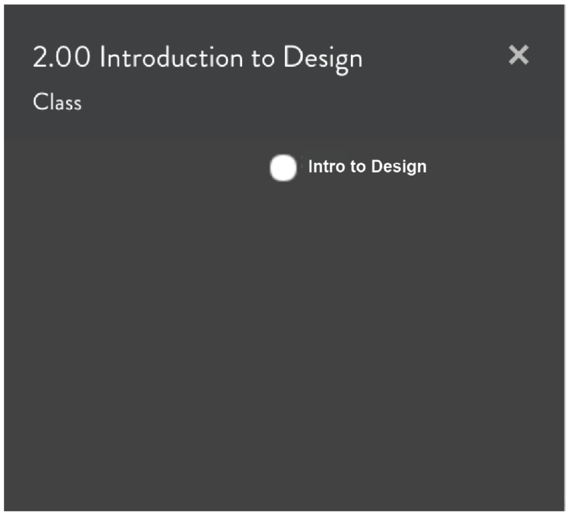
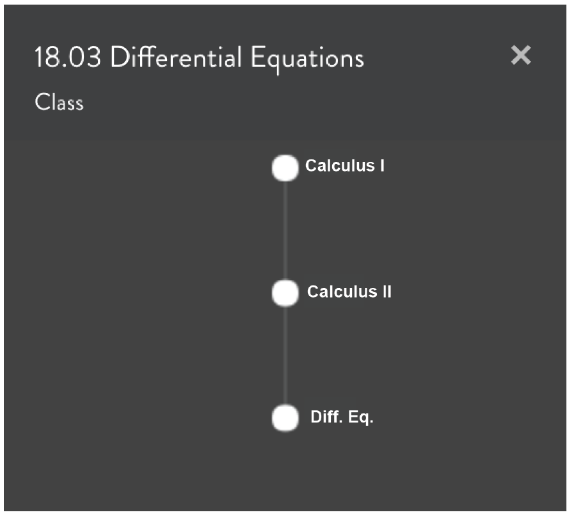

Search the map
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Science, Technology, & Society
Political Science
Biological Engineering
Chemical Engineering
Brain & Cognitive Sciences
Aeronautics & Astronautics
Media Arts & Sciences
Materials Science & Engineering
Anthropology
Women's & Gender Studies
Urban Studies & Planning
Electrical Engineering & Computer Science
Chemistry
Earth, Atmospheric, & Planetary Sciences
History
Health Sciences & Technology
Nuclear Science & Engineering
Physics
Mathematics
Mechanical Engineering
Civil & Environmental Engineering
Writing
Economics
Music
Biology
Global Studies & Languages
Management
Literature
Architecture
About this map
Mechanical Engineering
2.001 Mechanics and Materials I
2.05 Thermo-dynamics
2.051 Intro. to Heat Transfer

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.

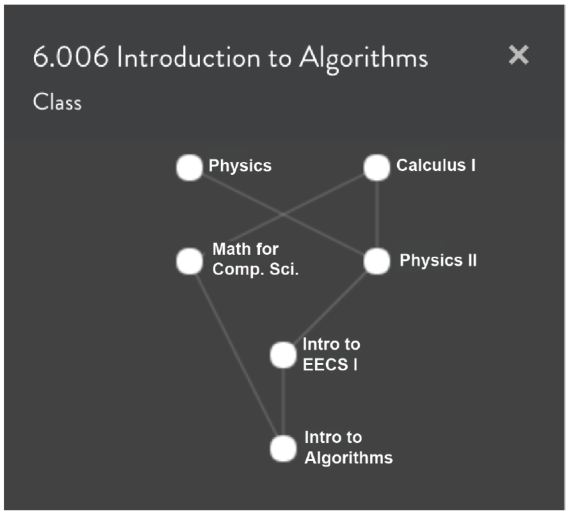
longest path that requires three levels of prerequisite courses. Note that in the interest of presenting a simple illustrative graphic, these visualizations do not attempt to represent "or" conditions in the prerequisite pathways.



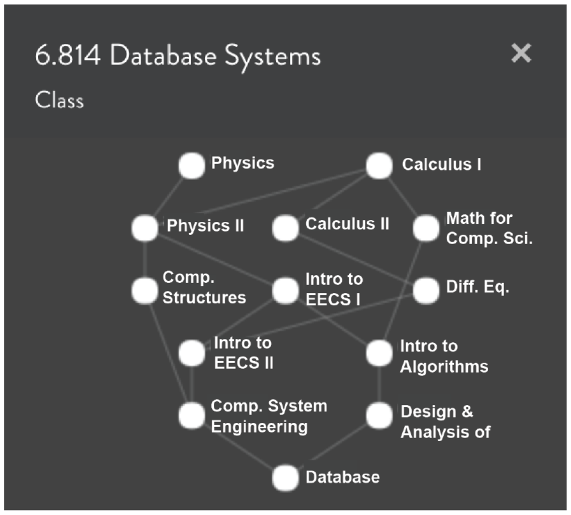
(a) The MIT course 2.00 Introduction to Design has a rank of 0.



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



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



(d) The MIT 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. In all cases the directionality of the edges between courses points downwards.

4.2 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.

4.2.1 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. In this case, the outcomes are specified by an external accreditation agency. 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). This list and the weights were created by the SUTD faculty and curriculum coordinators. 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.

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 whiter nodes addressing more outcomes and darker nodes addressing fewer 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	*Description*
Course	85	*Learning evidence URL*
Group	7	
Program	1	
Relationships	**Count**	**Attributes**
has-parent-of	132	*Type, Directionality*
addresses	1559	*Type, Directionality, Strength*

4.2.2 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

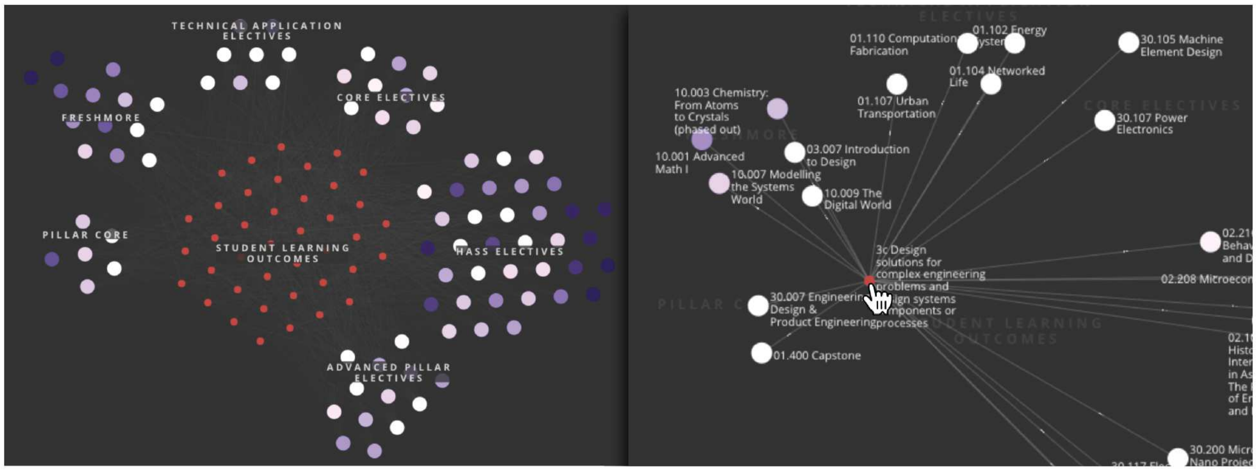

TECHNICAL APPLICATION ELECTIVES
CORE ELECTIVES
FRESHMORE
PILLAR CORE
STUDENT LEARNING OUTCOMES
HASS ELECTIVES
ADVANCED PILLAR ELECTIVES
01.110 Computation Fabrication
01.102 Energy System
30.105 Machine Element Design
01.104 Networked Life
10.003 Chemistry: From Atoms to Crystals (phased out)
01.107 Urban Transportation
30.107 Power Electronics
10.001 Advanced Math I
03.007 Introduction to Design
10.007 Modelling the Systems World
10.009 The Digital World
3c Design solutions for complex engineering problems and design systems components or processes
30.007 Engineering Design & Product Engineering
02.208 Microecon
01.400 Capstone
30.200 Micro Nano Projec

Figure 10: Visualization of SUTD Engineering Product Development degree program mapping. Outcome nodes are shown as small red circles and course nodes are shown as larger circles. Whiter course nodes address more outcomes and darker course nodes address fewer outcomes. The mouseover on the right shows which courses address the highlighted outcome.

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.

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

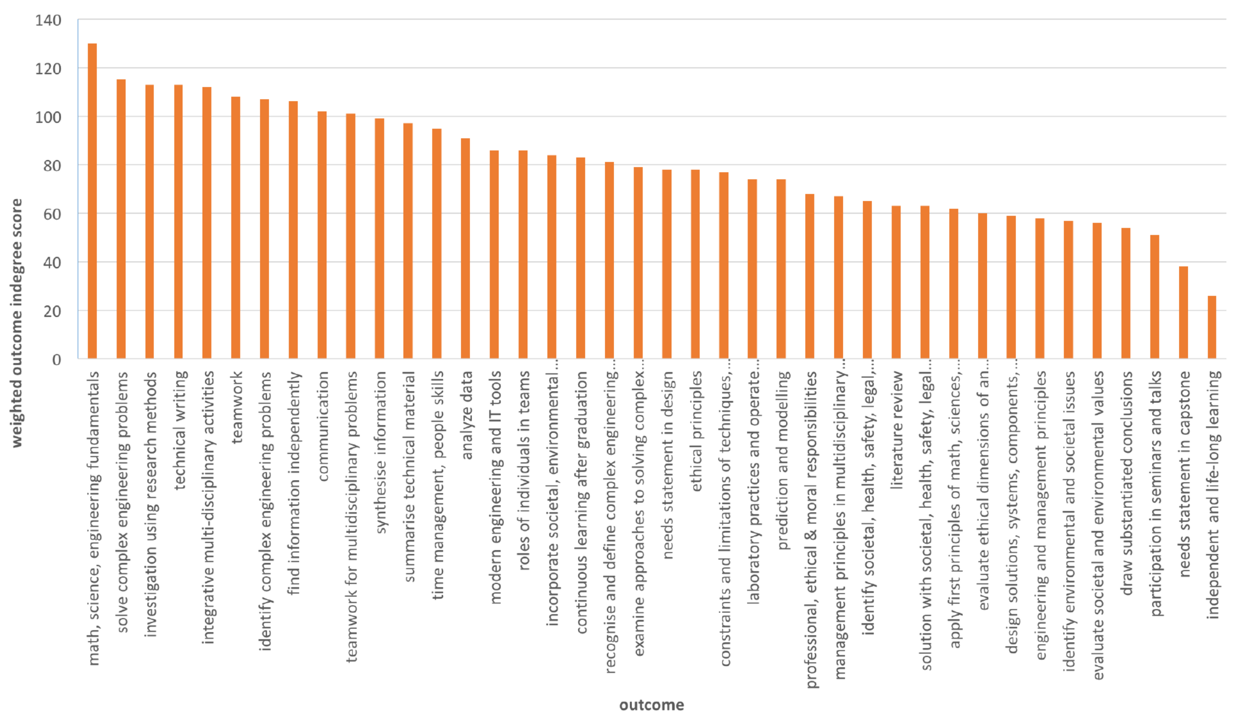


Figure 11: The weighted indegree score for each outcome shows how strongly the outcome is addressed by courses across the SUTD EPD curriculum.

also be combined with student enrollment data to provide a quantitative assessment of student coverage of outcomes, either by individual student or in aggregate across the student population.

To further illustrate the utility of this analysis, we create a notional student plan of study,[6] which identifies the courses taken or planned to be taken by a particular (notional) student who we term "Student X." We layer Student X's data on the accreditation network model, in order to analyze the strength of outcome coverage specifically for Student X, projected upon completion of their degree program. Figure 13 shows for this notional example the projected distribution of coverage for Student X. The red bars in Figure 13 highlight two outcomes that have low coverage scores: (1) Recognise the need for and prepare for independent and life-long learning and (2) Regular participation in seminars and talks. To remedy the deficiency in these two outcomes, the student would be advised to take a course that addresses these outcomes. With the network model in hand, we can immediately generate a recommendation for a course that strongly *addresses* both of these outcomes. Figure 14 shows a companion visualization that highlights this recommendation. In this case, the course *Sociology of Social Networks and Social Capital* is identified to be a course with one of the highest *addresses* edge values.

[6]Due to privacy concerns, real student data are not depicted here.

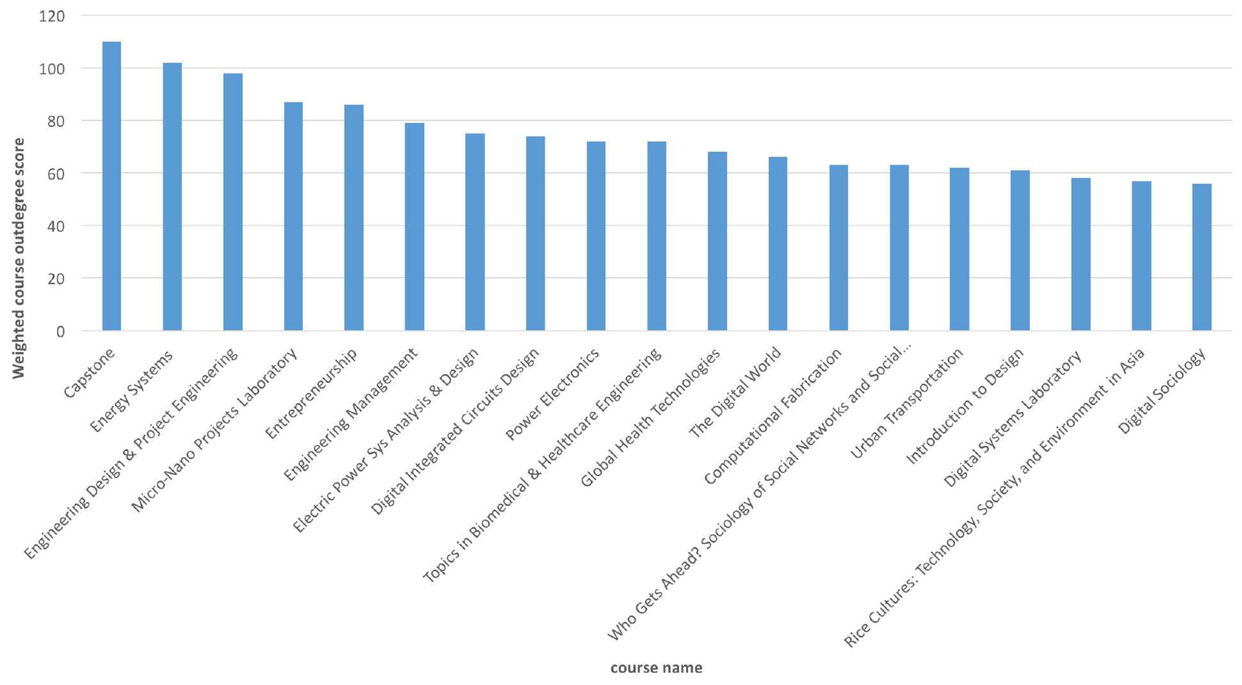


Figure 12: The weighted outdegree score for each course shows how strongly the course contributes to outcomes across the SUTD EPD curriculum.

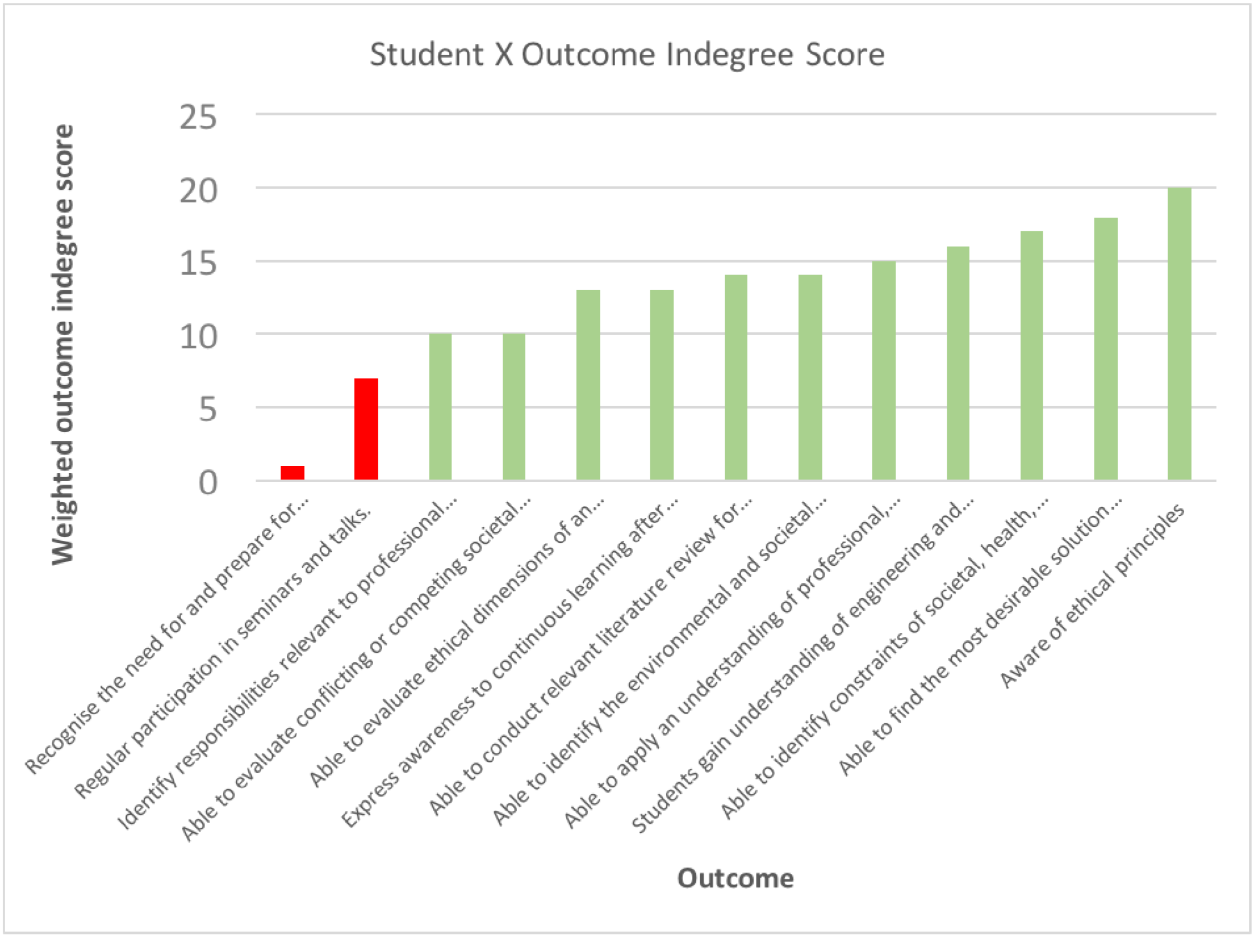


Figure 13: The projected outcome coverage of Student X, shown in increasing score. A red bar indicates that Student X has a potential deficiency for that outcome with the current plan of study.

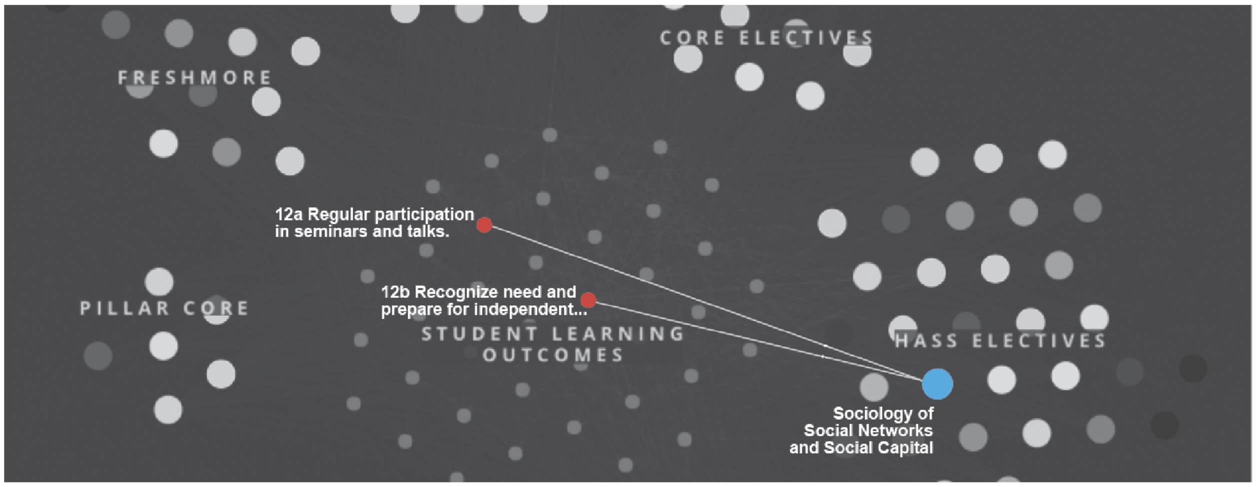

FRESHMORE
CORE ELECTIVES
PILLAR CORE
STUDENT LEARNING OUTCOMES
HASS ELECTIVES
12a Regular participation in seminars and talks.
12b Recognize need and prepare for independent...
Sociology of Social Networks and Social Capital

Figure 14: The projected outcome coverage of Student X was found to be deficient in two outcomes, highlighted in red. To alleviate this deficiency, the highlighted blue course that addresses both outcomes is recommended to the student in an advising session.

4.3 Modeling at the course scale

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

4.3.1 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 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 15 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	*Type, Directionality*
addresses	157	*Type, Directionality*
leads-to	50	*Type, Directionality*

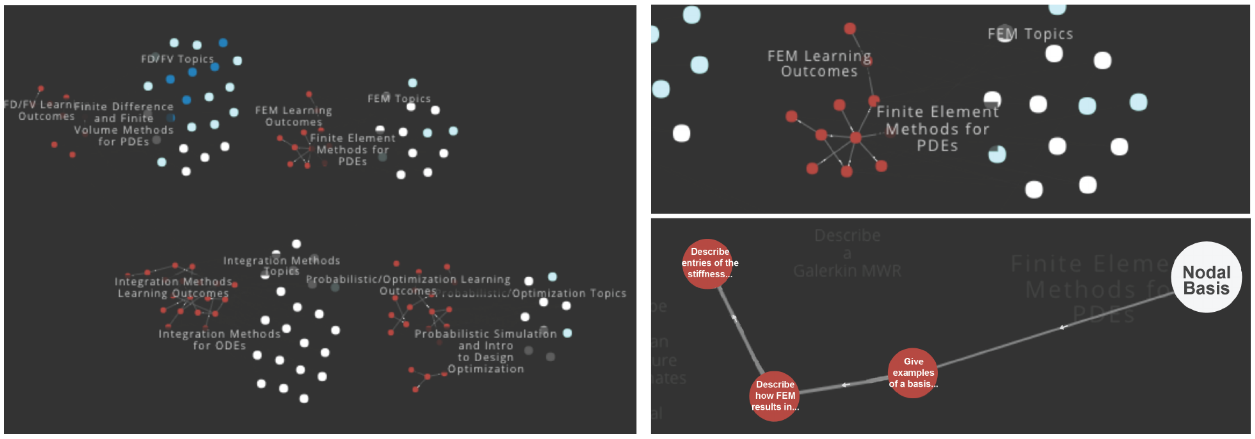

FD/FV Topics
FD/FV Learning Outcomes
Finite Difference and Finite Volume Methods for PDEs
FEM Learning Outcomes
FEM Topics
Finite Element Methods for PDEs
Integration Methods Topics
Integration Methods Learning Outcomes
Integration Methods for ODEs
Probabilistic/Optimization Learning Outcomes
Probabilistic/Optimization Topics
Probabilistic Simulation and Intro to Design Optimization
Describe entries of the stiffness...
Describe a Galerkin MWR
Describe how FEM results in...
Give examples of a basis...
Nodal Basis

Figure 15: Visualization of concept map for course Computational Methods for Aerospace Engineering.

4.3.2 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. The three outcomes of rank four are shown in Table 7. This means that, for those three outcomes with rank four, there is at least one path that includes four prerequisite outcomes that *lead to* that outcome. Note that here it is the longest path that is of interest, because this represents the most constraining (in terms of number of prerequisite outcomes) requirement in the concept map pathways. The table also shows the total number of prerequisite outcomes for each outcome (including 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 16 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. From the figure it can be seen that there are additional shorter pathways leading to this node (e.g., the pathway with just two prerequisite outcomes that flows through "Explain the Newton-Raphson method"); however, since a student must master *all* upstream outcomes, it is the longest pathway that is relevant in ranking the outcome.

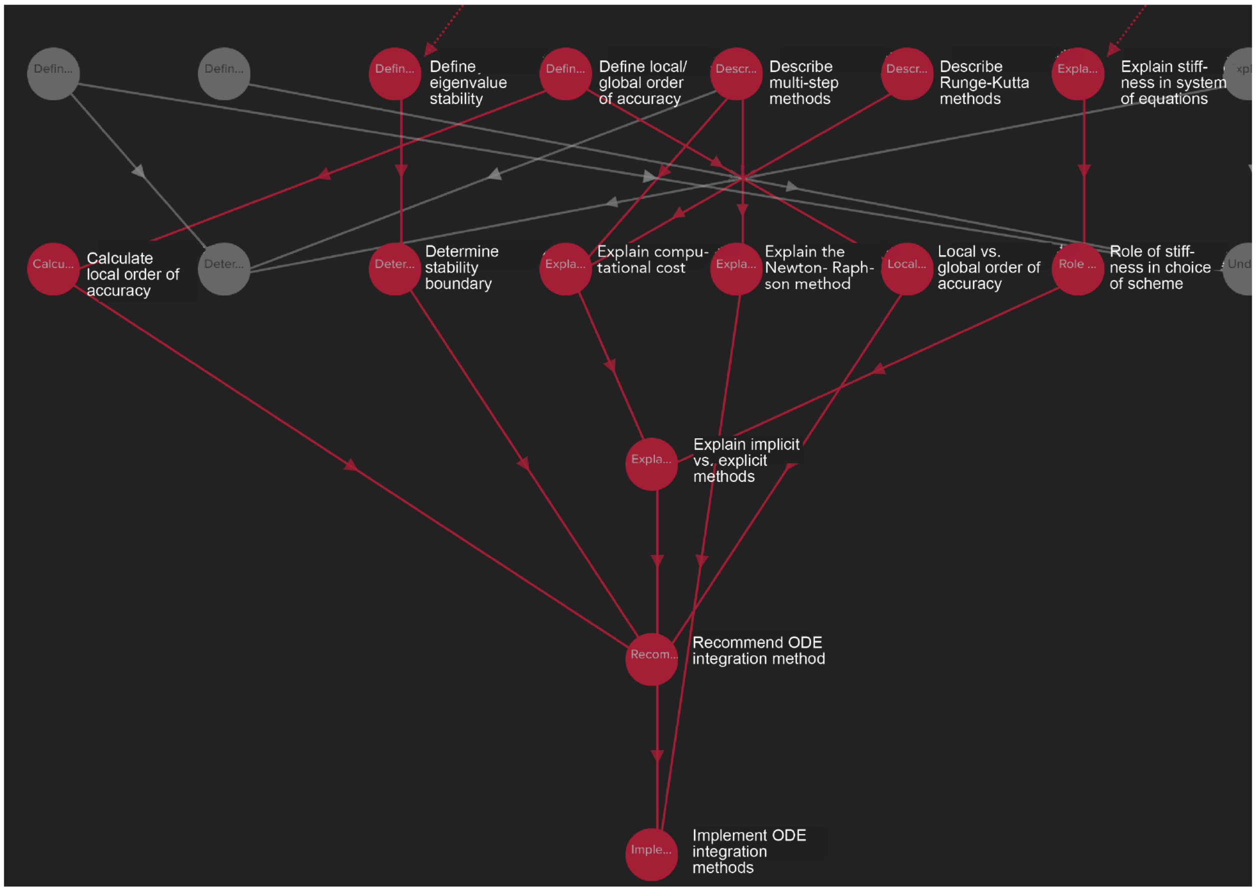

Defin...
Defin...
Defin...
Define eigenvalue stability
Defin...
Define local/ global order of accuracy
Descr...
Describe multi-step methods
Descr...
Describe Runge-Kutta methods
Expla...
Explain stiff-ness in system of equations
Calcu...
Calculate local order of accuracy
Deter...
Deter...
Determine stability boundary
Expla...
Explain compu-tational cost
Expla...
Explain the Newton- Raph-son method
Local...
Local vs. global order of accuracy
Role ...
Role of stiff-ness in choice of scheme
Und
Expla...
Explain implicit vs. explicit methods
Recom...
Recommend ODE integration method
Imple...
Implement ODE integration methods

Figure 16: Visualization of the tree-like structure yielded by the outcome *Implement ODE integration methods* and its prerequisite chain.

5 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, student course planning and student advising.

It is important to note that in many cases the types of analyses presented in this paper depend critically on data provided by instructors and faculty. For example, in the accreditation mapping example, instructors provide the data that specifies which accreditation outcomes are addressed by their courses and to what extent (via the weightings). Similarly, in the concept mapping example, the instructor provided the learning outcomes and defined their relationships to the concepts. The conclusions drawn from the analysis methods presented in this paper will only be as good as the

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

data on which they are built, although in this regard scalable visualization can be a valuable way of communicating and checking data.

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 data-driven advising, 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. This work was supported in part by the International Design Center as part of the MIT-SUTD Collaboration.

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] H. Jacobs. *Curriculum 21: Essential Education for a Changing World*. ASCD, 2010.

[14] 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.

[15] K. Kelley, J. McAuley, L. Wallace, and S. Frank. Curricular mapping: Process and product. *American Journal of Pharmaceutical Education*, 72(5), September 2008.

[16] K. Koedinger, A. Corbett, and C. Perfetti. The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36:757–798, 2012.

[17] 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.

[18] J. Novak and D. Gowin. *Learning How to Learn*. Cambridge University Press, 1984.

[19] 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.

[20] 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.

[21] B. Saxberg. Learning engineering: The art of applying learning science at scale. *L@S '17 Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, April 2017.

[22] M. Scott. *Programming Language Pragmatics*. Morgan Kaufmann Publishers, San Francisco, CA, 2006.

[23] 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.

[24] G. Siemens and P. Long. Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, 46(5):30–40, 2011.

[25] J. Turns, C. J. Atman, and R. Adams. Concept maps for engineering education: A cognitively motivated tool supporting varied assessment functions. *IEEE Transations on Education*, 43(2):164–173, May 2000.

[26] D. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, 2001.