Graph-theoretic approaches and tools for quantitatively assessing curricula coherence

***Abstract*—In this paper, we propose a method to analyse the coherence of existing curricula at higher education institution. We focus our attention to engineering programs at universities but the proposed method is by no means restricted to those cases. In contrast to other known methods, our approach is quantitative, decentralised, asynchronous, allows to analyse entire programs (in contrast to single courses) and does not depend on using specific teaching methods or tools.**

We propose to perform this quantitative assessment in two steps: first, representing the university program as an opportune graph with courses and concepts as nodes and connections between courses and concepts as edges; second, analysing the structure of the program using methods from graph theory.

We thus perform two investigations, both leveraging a practical case – data collected from three engineering programs at two Swedish universities: a) how to represent university programs in terms of graphs (here called Concepts-Courses Graph (CCG)), and b) how to reinterpret the most classical graph-theoretical node centrality indexes and connectivity and network flow results in order to analyse the program structure, including to discover flows and mismatches.

***Index Terms*—Concepts-Courses Matrix (CCM), Concepts-Courses Graph (CCG), university program design, graph theory, centrality, connectivity**

I. Introduction

A. Background

Developing curricula is a complex task that involves creating and assessing proposals in different organizational and decision contexts. There is a significant body of literature dealing with models of curriculum design, e.g., [20]. As reported in [10], two established models dominate: the *Objectives Model*, that starts by specifying the objectives or learning outcomes defined as measurable performances, see also [3], and the *Process Model*, that starts by defining course content and specifying criteria to assess students' knowledge of this content. Several variations on these models exist (e.g., Tyler's, Wheeler's, and Kerr's Models).

When it comes to the individual courses in a program, a new curriculum might comprise courses that already existed for example in a different but similar program, and new courses that are designed for the specific curriculum. Deciding how to segment the material into existing or new courses often follows a discipline-centered approach. This is likely due to the ease of practical implementation given numerous books and support material available to aid course designers, e.g., [15, 4, 19].

It is well known that understanding how knowledge within a course or across courses in a program is conceptualised can provide a clearer basis for creating a structure and progression that better supports student learning. Towards reaching this goal, a well known strategy is the so-called *black-box* approach to the sequencing of a curriculum [5]. This development tool has been proposed as part of the Conceiving, Designing, Implementing, and Operating (CDIO) standard for the management of university programs and entails representing every course within a program as a set of inputs (e.g., prerequisite skills and knowledge) and outputs (e.g., contributions to the final learning outcomes). Coupling this information with all courses is expected to enable discussions, make connections (or lack thereof) visible, provide an overview of the program, and eventually serve as a basis for both planning and improving. However, this tool is still qualitative, and does not provide quantitative indications that are not subject to personal interpretations.

B. Curricula Management

Curricula management is usually required during the entire life span of a university program for several important reasons. Even in the best designed and aligned programs, courses may be moved within the curriculum due to pragmatical needs (e.g., they do not fit in the schedule from logistical reasons, or there is no teacher available) leading to changes in how and which knowledge can be or is taught. Further, the profiles of the teachers can, consciously or unconsciously, change the curricula over time at least gradually.

At the same time, higher education institutions strive to more heavily integrate research-based practices at all levels from teaching to program design and development. In fact, as pointed out in [18, p. 6], this institution-level transformation is important, and should be reflected in opportune transformations of the university programs. However, the vagueness and lack of facility to objectively measure the goals of higher education may lead faculty members to realize and prioritize these goals based on their own interpretations [6, 17]. Indeed, this may lead to decisions that are heavily steered by charismatic people with strong opinions, or affected by board members' lack of time/interest, or by communication difficulties, especially when program design and management involves cooperation between academics from different disciplines or traditions.

Yet another problem is that a program board cannot have full control of all parts of a program, much less on how a course is examined. It is also the case that some goals may be implicit, unspoken or simply assumed obvious. Moreover, in many cases it is not clear what a specific course shall deliver in form of decided (or intended) knowledge accordingly, or what is its intended connection to other courses and content of the program.

Hence, in summary we identify the following shortcomings in the common practise of program boards to maintain coherence within a curriculum or part of it: (i) this operation is typically done centrally (by the program board), which usually cannot collect or use all relevant information; (ii) it involves collecting, revising and reviewing information manually, which limits how much data can be processed, and (iii) it is further done at discrete time instances / according to fixed cycles, and this may delay or prevent the detection of mismatches (specially if long temporal delays are present between the implementation of local changes in courses' contents and their theoretical reporting in the next revision of the program).

In some cases detailed information about a course and its contents and detailed data are available, specially if the courses are taught through Intelligent Tutoring Systems (ITSs), which record when students passed or failed quizzes, accessed certain material and many other aspects related to student behaviour and learning. The insights are then often used to model the students and derive strategies to better adapt the teaching experience to the specific student's needs. The tools are particularly useful when teaching large classes such as in MOOCs. However, they are not often used in traditional university programs since it requires to transform the course such that it can be taught using an online tool (often requiring a complete new set of materials) and having a suitable electronic tool available, which might not be affordable or desirable for all universities. However, since ITSs are not widely used in standard university programs, a curriculum maintenance system should not depend on data from such electronic systems. Moreover, the purpose of ITSs is to adapt teaching material and learning strategies to individual students. In contrast, a curriculum maintenance system aims at improving the quality of curricula coherence and maintenance efforts and should also be applicable to any university context, independently of which teaching method is used in the different courses.

C. Introducing Quantitative Management

In this paper, we suggest a method that allows for building a "curriculum monitoring system" which is expected to improve the current method outlined above by being:

- decentralised and collaborative, by directly involving primarily the teachers and possibly also students in the program (a component that also increases the probability that they will accept and act upon the results),
- continuous and asynchronous, in the sense enabling to assess the coherence of the program, curriculum or part of it at any time (i.e., not necessarily being tied to fixed time instances) and as soon as some new data are available (i.e., asynchronous as not needing the collection of every data from every courses to already start providing actionable information),
- flexible, in the sense that allows to include courses taught with a wide range of teaching and learning methods,
- data driven, in the sense of allowing to analyse the curriculum, its knowledge flow and its coherence with computer aided objective tools (compared to summarising and revising the information manually, potentially including more data than humans can usually handle).

We expect that quantitative methods that can complement classical discussion-based approaches should – at least intuitively – lead to more sustainable and efficient management of university programs, and lead to the other ancillary benefits described in the statement of contributions below.

We propose a strategy that makes use of concepts from graph theory, such as analysing the node centrality, connectivity and flow in a network that describes how concepts and courses are related within a program. Our work relates thus to existing works that consider graph theoretic approaches for the management of curricula studies.

An example of such a method is the one summarized in [14], where the author collected in her PhD thesis an extensive approach to perform a quantitative analysis of the structure of curricula by means of four distinct graphs that could be used to create a study plan for students. The objectives of her body of work are on helping teachers and students create personalized study plans for individualized students learning through graph theory. This means that there is no explicit focus on how to help teachers understand how to compile these graphs, collaborate to compile them, and use the results to foster alignment among the stakeholders.

The approach presented in [1] was used to analyze connections between courses according to curriculum structure in order to understand the coherence and structure within a university program. This work focuses on defining and analysing the network of prerequisites of the various courses, with the purpose of understanding how their roles can differ within a curriculum. In other words, the work focuses only tangentially on how to use the information provided by these networks for taking actions. Moreover, the important questions of *why* courses are prerequisites for other courses and *what is learned* in the courses is not considered here.

Another paper connecting learning goals (intended as desired results), topics, and courses within a program is [13]. Here the authors model learning goals, topics and courses as nodes, and model prerequisite dependencies as edges, so that the relation between courses and topics are represented as edges. Graph analysis techniques are then utilized to measure several aspects of a curriculum; the overall information is then used to perform automated curriculum design operations (e.g., to automatically generate the draft of course syllabi). It is important to note that the focus of this paper is more on the design of curricula, rather than on estimating the current situation. Moreover the described approaches do not include procedural information explaining who does what from a practical perspective (in a sense, not giving ”instructions” on how to practically replicate the approaches in other universities.

We also report [12], that proposes to use graphs-oriented approaches to answer key questions on where to focus the assessments, data collection, and corrective actions within the curriculum. The focus here is thus on how to analyze systematically and quantitatively the program contents so to help faculty and administrators that are tasked with creating quality assurance and assessment schemes.

With respect to the graph-theoretic approaches for representing university program contents discussed above, we focus on implementing a collaborative and team-based strategy for obtaining graph-oriented representations that can help aligning expectations and language specially among the teachers, and between teachers and students. In a sense, we are more concerned with creating conditions that favor the implementation of constructive alignment paradigms. We thus devise a strategy that is a bottom-up approach to obtaining graph-oriented representations of university programs.

Structure of the manuscript: Section II describes the tools for collecting and representing quantitative information about a generic university program. Section III discusses how classical node centrality indexes and graph connectivity can be interpreted and applied in our context of university programs analysis. Section IV reports and examines the results obtained from field applications of the proposed method. Finally, Section V presents some concluding remarks and suggests some future research and development efforts.

*

II. The Concepts-Courses Matrix and the Concepts-Courses Graph

To build a tool that teachers can use in a bottom-up fashion to quantitatively evaluate and analyze the structure of the university programs related to their courses, we exploit the intuition that courses within a program are connected through a flow of concepts that are taught throughout the duration of a given program. This intuition is common especially in engineering disciplines, where constructivist interpretations of knowledge tend to prevail.

TABLE I: Part of a Concepts-Courses Matrix taken from the field case of Electrical Engineering, academic year 2017 / 2018, Uppsala University, Sweden.

	1TE705	1TE704	1MA008	1TE667
	Intro to El. Eng.	Components & Circuits	Algebra & Vector Geom.	El. Circ. Theory
complex num.			2	2
vectors			2	1
sys. of lin. eq.			2	2
Ohm’s law		2		2
Kirchoff’s laws		2		2
pot. voltage		2		2
linearity				
matrices	1	1	2	2
work, energy		2		2
int. calculus	1			

To highlight the connections among courses, we suggest to follow a procedure based on executing two separate steps: *acquire the data*, as described in Section II-A, and *visualize the data*, as described in Section II-B.

A. Data acquisition through the Concepts-Courses Matrix tool

The simplest teachers collaboration strategy that we devise is to let them define a table, in the following denoted as Concepts-Courses Matrix (CCM), where the columns / rows correspond to the courses / concepts within the program (see Table I for an example). A CCM thus allocates one column per course j and one row per concept k, so that the value of each (k, j)-th element may be used to quantify how relevant the concept k is for course j on a predefined scale. As for which scale to use, as indicated in [9] a simple option consists in “0” = not relevant, “1” = somewhat relevant but not central, and “2” = very relevant / central for the course. A more refined option may instead consist in associating a number to each level of an opportune taxonomy (e.g., Bloom’s), and then let each (k, j)-th element in the CCM be either 0, if concept k is not related to course j, or the number corresponding to the learning level that students should ideally reach about the k-th concept when they successfully conclude course j. Note that, despite being more informational, using taxonomy levels would require all the collaborating teachers that define these tables to be acquainted with the concept of taxonomy and share the same interpretation of the meanings of the various levels.

Building a complete CCM for a specific program or part of it at a specific department requires the persons collaborating on the creation of the CCM executing two steps: *a)* defining which concepts shall be included, and *b)* interviewing (also through internet-based tools) experts that may give indications on the values of the elements in the matrix.

As for step *a)*, a natural strategy is to first build an initial list of important knowledge by inspecting each course description in the program (or part of it), and receive feedback on this from the board and the teachers involved in the program (or part of it). A different option may be to consider which questions are asked in the various exams of the various courses.

As for step *b)*, one may exploit several possibilities:

Inputs from the teachers: One option is to collect relevant data for each individual course from the teacher teaching that specific course, as she/he can be regarded as an expert on her/his particular subject (so that she/he can often produce such information with ease). However, in our experience we noticed that this strategy has several disadvantages. First, teachers might be unwilling to spend time on preparing such data, especially when the CCM comprises a long list of concepts, or if they do not see immediate benefits from doing so. Further, such information will inevitably reflect what each teacher thinks or desires to teach in her/his course. This information might be quite different from the perception of both students and other teachers or the effects actually achieved in terms of what students learn. Strategies to mitigate these subjective distortions are proposed and analyzed in Section IV below.

Inputs from the students: To complement and validate the inputs from the teachers we consider also asking students to provide information on the various courses that they have been taking. This information may however be distorted, too. For instance, students might not always understand how concepts, facts and other knowledge are interrelated or which knowledge is a prerequisite for a given course. To do so, a metacognitive understanding of the course is required, but in our experience this does not happen for all the students and often only after some time of reflection. We expect that some of these issues can be resolved or attenuated by averaging over the data from several students. One might also consider combining the information from students with inputs from teachers or other sources. But great care should be taken when intending to do so. Indeed, despite asking the same question to students and teachers, namely, which concepts were required or developed by a given course, their answers will differ due to their perspectives. Teachers will likely answer according to what they intend to teach, also influenced by their usually much wider metacognical understanding and reflection of the material. In contrast, students will answer according how they understood and perceived the course. In fact, rather than averaging, comparing the data from teachers and students might be a more viable tool to detect possible mismatches between teachers' intends and students' perceptions.

Analysis of the exam questions: Another option (not explored in our field tests) is to inspect which questions are asked in the exams of the various courses, and link those to the list of concepts created in step *a)*. This information would also complement the teacher input about her/his Intended Learning Outcomes (ILOs). This approach might be particularly useful at European universities, since according to the Bologna process, see for example [8], exam questions should be closely connected to, and examine the learning outcomes, i.e. course goals. Hence, these may be a valuable source to categorize the ILOs as well as the course goals.

Formal sources: As another option, formal sources such as analysing course goals or interviewing program boards or study directors might reveal information about which requirements higher authorities intend a course to fulfil. However, in some cases, this might be wishful thinking or it might be difficult or impossible to extract relevant information as authorities might not have sufficient insights into all courses. Further, course goals might be formulated unsuitably.

Note that all information sources listed above lead to subjective data. Indeed, objective data on which concepts are included in which course do not exist. However, in order to minimize the influence of subjective opinions, several sources of information should be combined weighing the data in an appropriate manner and considering the possible bias of each source. In our study we followed mainly the option "*inputs from the teachers*".

B. Data visualization through the Concepts-Courses Graph tool

After obtaining the CCM described above, the program can be represented as an undirected weighted bipartite[1] graph, denoted as a Concepts-Courses Graph (CCG). The two sets of nodes in this graph correspond to the courses k and the concepts j within the program. The (k, j)-element in the CCM corresponds then to the weight of the edge between the concept node k and the course node j (e.g., see the example in Figure 1). Intuitively, the properties of a university program (e.g., its structure, the relations and the relevance of the courses and concepts in a program, the existence of potential flaws in its design) should translate into opportune topological properties of the CCG. If this intuition is true, then the problem of quantitatively analyzing a university program can be cast as the problem of analyzing the opportune graph. The problem of understanding what can actually be inferred about a university program through analyses of its CCG is discussed in Sections III and IV.

C. Extending the CCM and CCG tools to distinguish prerequisite information from taught information

The CCM and CCG tools defined and described above may be structured in a more complex way so to include more information. For example, one major shortcoming of the CCM and CCG tools above is that they do not embed why a concept is relevant for a course. For instance,

[1]In the sense that its nodes can be divided into two disjoint independent sets such that all edges connect a node from one of these sets to the other. We moreover here use "network" in the sense of a collection of agents and connections between them which can be represented as a graph. For definitions and a comprehensive study of graphs and graph theory, please refer to [7].

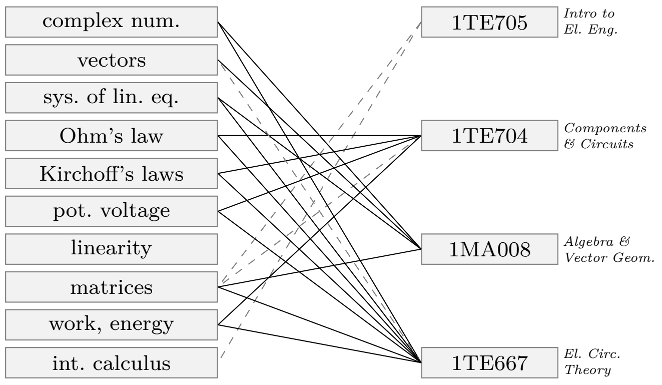

Fig. 1: Example of a CCG corresponding to the CCM in Table I.

TABLE II: Example of part of a Directed Concepts-Courses Matrix taken from the field case of Computer and Information Engineering, academic year 2017 / 2018, Uppsala University, Sweden. (T = 'taught' in the course, R = 'required' by the course)

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

a concept could be relevant because it is a necessary prerequisite to fruitfully follow a course, or because it is an ILO developed and taught in the course. Collecting this type of information allows more detailed analyses. To enable these, however, one has to let the CCM have two columns for each course: one allocated for weights to quantify the relevance of prerequisite concepts (e.g., the learning levels that students should ideally have to be able to follow fruitfully the course), and the other one allocated for weights of concepts taught or developed in that course (e.g., the learning levels that students should ideally reach about that concept after having passed the course). Alternatively, one may indicate with an opportune symbol whether a specific concept is required or taught by a specific course. A field example is provided in Table II.

Correspondingly, one may extend the CCG defined in Section II-B by letting it become a directed, weighted, bipartite graph where edges that are directed from concept nodes to course nodes indicate that the concepts are required for that course, while edges that are directed from course nodes to concept nodes indicate that the concepts are taught by that course. The resulting graph, denoted as a Directed Concepts-Courses Graph (DCCG), would then look like in Figure 2. Note that combining this additional

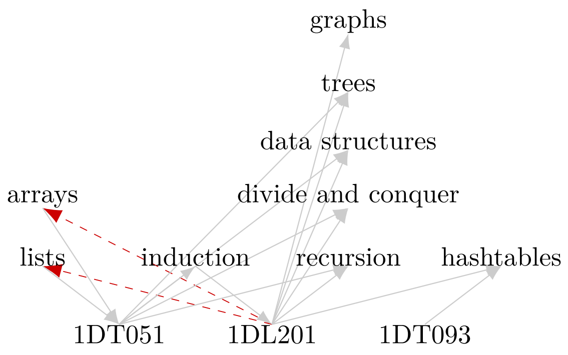

Fig. 2: Example of a DCCG corresponding to the DCCM in Table II, where for simplicity the weights of the edges have been omitted, and where moreover the course nodes are sorted temporally ascending from left (i.e., earlier courses) to right (i.e., latter courses). The dashed red arrows indicate concepts that are considered prerequisites for a given course, but taught only by following courses.

information on what are requirements and what are course outcomes can be used to, e.g., discover when early courses treat a given concept as prior knowledge despite it being only introduced in a later course. This occurrence is for example highlighted with red arrows in the field-example of Figure 2.

Our experience indicates that great care should be given on how to graphically represent a DCCM, since it constitutes a representation that can be very informative for the stakeholders (i.e., boards, teachers, and students). Up to now, our choice has been to follow these guidelines:

- plot course nodes on the bottom, in a temporally ascending order;
- plot concept nodes in vertical piles that lie between course nodes, and let the positions of these piles indicate whether concepts are prerequisites or teaching outcomes of the various courses.

More specifically, we propose to divide concepts in two sets: *a)* the "problematic" ones, i.e., that concepts for which there exists some early course that treats them as prior knowledge despite they are being introduced only in a later course (like the ones reached by the dashed red arrows in Figure 2), and *b)* the non-problematic ones. Given a non-problematic concept, we place it:

- just before the first course that has it as a prerequisite, if such a course exists (e.g., "induction" in Figure 2);
- otherwise, if no courses has that concept as a prerequisite, then immediately after the last course that teaches it (e.g., "recursion" in Figure 2).

As for problematic concepts, such as "arrays" and "lists" in Figure 2, our proposal is to plot them in the position they would be if we were ignoring the edges that make them problematic. In our experience this strategy enables seeing the whole program as a temporal flow and helps discussions on the program structure. Other representations are possible, but at least in our limited experience the proposed one has been the most beneficial for the purpose of aiding constructive meetings.

III. Data analysis methods

In this section we assume to have collected enough information so that, for a given university program, both the relative CCM described in Section II-A and the corresponding CCG in Section II-B have been compiled. Due to the special structure of these tools (i.e., a matrix and a graph), one can cast the problem of analyzing the properties of the program into the problem of analyzing the properties of the corresponding matrix or graph by means of well known and established tools from matrix and graph theory. Our purpose is then to discuss what these established tools say about potential structural problems of the represented programs. The section is divided in three main parts: Section III-A, discussing the pedagogical meaning of several nodes centrality measures; Section III-B, discussing the pedagogical meaning of standard network connectivity measures; and Section III-C, discussing the pedagogical meaning of cycles within a directed CCG.

A. Centrality measures

It can be revealing to understand how "important" or "central" certain courses and concepts are in a program, since this may give indications on which courses and concepts should receive special attention (e.g., in the form of additional students learning monitoring actions). To this aim, we notice that several well-established node centrality indexes exist in the literature (see, e.g., [7]).

Formally, we thus let the CCG be defined as the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, S\}$ is the set of nodes composing the graph and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of the edges between the nodes. To every edge $(i, j) \in \mathcal{E}$ corresponds an associated edge weight $w_{ij} \geq 0$. Since we are considering the situation described in Section II-B, $\mathcal{G}$ is static (i.e., not time-varying) and undirected (i.e., $(i, j) \in \mathcal{E} \Leftrightarrow (j, i) \in \mathcal{E}$, $w_{ij} = w_{ji}$). Given this, the set of neighbors of a generic node i is defined as $\mathcal{N}_i := \{j \mid (i, j) \in \mathcal{E}\}$. Finally note that $\mathcal{V}$ is bipartite, i.e., $\mathcal{V} = \mathcal{V}_{\text{courses}} \cup \mathcal{V}_{\text{concepts}}$, and $(i, j) \in \mathcal{E}$ must be so that either $i \in \mathcal{V}_{\text{courses}}$ and $j \in \mathcal{V}_{\text{concepts}}$ or vice versa.

1) Degree centrality: This centrality index is one of the simplest ones, being the sum of the weights of all edges connected to a particular node, i.e.,

$$d(i) = \sum_{j \in \mathcal{N}_i} w_{ij}. \tag{1}$$

In our pedagogical-oriented interpretation of $\mathcal{G}$, the degree-centrality indicates the weighted sum of how many courses a particular concept is relevant (connected to) or how many concepts are taught or are connected to a particular course with the weight being related to the importance on the scale $\{0, 1, 2\}$. However, this metric is very sensitive to how the connections are weighted. For instance, consider that, to compile the CCM as we indicated in Section II-A, teachers have to add "1"s or "2"s to the concepts relevant to their courses. As we noticed in our field tests, different teachers have different compilation approaches (i.e., more conservative teachers might think that the compilation should focus just on core concepts, and hence assign less overall weights than other teachers that assume that the compilation should be "exhaustive") . What we noticed, thus, is that the degree centrality scores are very sensitive to the personality of the various teachers, which is not a desirable property. In order to compensate for this effect, one may then think that weights can be adjusted or normalised; however, normalising weights so that the accumulated weights of each course add up to an assigned number (e.g., the credit points associated to that course or simply 1), then the degree will inevitably be this number, so that the metric loses its value. Hence, weights should be chosen with great care and establishing a common understanding among teachers on how to assign weights is essential to retrieve meaningful insights from this specific metric.

2) Closeness centrality: This metric is intuitively defined as the average length of the shortest path between a specific node and all other nodes in the graph. Formally, for a connected graph, it corresponds to

$$c(i) = \frac{1}{\sum_j \operatorname{dist}(i, j)} \tag{2}$$

where $\operatorname{dist}(i, j)$ is the (geodesic) distance between node i and j, which is the length of a shortest path between the nodes and setting the length of the edge from k to l to $\frac{1}{w_{kl}}$, see also [7]. Closeness roughly expresses how close the particular node is to the remaining nodes of the network. When used in our context, a pedagogical interpretation may be how logically "far" a certain course or concept is from other courses or concepts. In other words, if i is a course node, and i has a very high closeness index, then it means that it is focusing on concepts that are used by several other courses. When i is a concept node, this index corresponds to how diverse the related concepts are, e.g., if they span over a large group of different subareas of the overall concept repertoire or not. When used for concepts, we expect this metric to be high for concepts that are taught or brought up in courses whose contents span over the entire program. Consequently, concepts that are only taken up by some courses or only during some time periods of the program are expected to be less close.

3) Eigenvector centrality (a.k.a. eigencentrality): This index assigns relative scores to all nodes in the network based on the idea that connections to high-scoring nodes contribute more to the score of a particular node than connections to low-scoring nodes with the same weights. Formally, the measure is defined as

$$e(i) = \frac{1}{\lambda} \sum_{j \in \mathcal{N}(i)} w_{ij} e(j) \tag{3}$$

where λ is a constant and turns out to be the largest eigenvalue of the adjacency matrix. A possible pedagogical interpretation for this measure is a quantification of how *influential* a course or concept is within the program. We expect this metric to give meaningful insights into the program structure for two main reasons: First, it is likely to be high for courses that cover many concepts that are also (very) relevant in (many) other courses. Second, this more complex measure goes well beyond what can be achieved by simply adding weights or manual analysis of the data, and is expected to be less sensitive to the teachers' personal interpretations of how to compile the CCM. Hence, we expect this measure to offer insights that can complement the ones that more intuitive and simpler metrics may give.

4) PageRank centrality: This is an adaptation of the eigenvector centrality discussed above, which assigns different scaling factors to the edges. More precisely, it is defined as

$$p(i) = \alpha \sum_{j \in N(i)} w_{ij} \frac{p(j)}{\sum_{k \in \mathcal{N}_j} w_{jk}} + \frac{1-\alpha}{N} \tag{4}$$

where the attenuation factor satisfies $\alpha \in (0,1)$. Due to the similarities with the eigencentrality, we expect the results from using this metric to be of similar usefulness. A discussion on the practical differences that we notice using the two metrics is given in Section IV.

5) Betweenness centrality: The graph-oriented interpretation of this centrality index is the one of a measure of how often the node acts as a "bridge" along the shortest paths between two any other nodes. Formally, the metric is defined as

$$b(i) = \sum_{j \neq i \neq k \in \mathcal{V}} \frac{\sigma_{jk}(i)}{\sigma_{jk}} \tag{5}$$

where σ_{jk} denotes the total number of shortest paths between nodes j and k and $\sigma_{jk}(i)$ is the number of those, that pass through node i. In our pedagogical setting this classical index, however, seems to be of limited importance. The CCG is indeed by design a bipartite graph, where concepts are exclusively connected to courses and vice versa. Hence, betweenness metrics are expected to be low for almost all nodes in the graph. This implies that the index might not discriminate among different nodes, and hence might not provide strong insights into the properties of a program.

B. Connectivity and network flow analysis

To complement the centrality measures presented above it can be useful to infer how courses and concepts are connected among themselves: this may give quantitative indications on how well connected the program is, which courses / concepts complement each other, and plan further learning / teaching monitoring & assessment activities. To analyse how a graph as a whole is connected there exist several well-established tools; among others, we discuss the pedagogical meaning of the connectivity, (minimal) cuts and the network flow concepts.

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

For instance, if a CCG happens to be disconnected, it reveals that different parts of the university program have no overlap or connection. This might indicate that concepts and courses of very different areas are included in the program, or that the program design may have some flaws. Hence, the teachers collaborating in the definition of the CCG might take this as a starting point to investigate a possibly undesired compartmentalisation of the program by investigating the (weakly) connected components of the graph.

We also note that a well designed curriculum will most likely contain some weakly connected DCCG. Indeed it is reasonable to expect that there will be pairs of nodes i and j for which there exists no directed path from i to j or viceversa. As an example, consider two courses that are given in parallel, which have some shared prerequisite concepts and possibly some shared developed concepts. Hence, if we ignore the direction of the edges, there exists a path between the two courses. However, as the information flows from the prerequisites through the courses to the developed concepts in parallel, there exists no directed path between the two courses. The same conclusion, i.e., that the DCCG cannot be connected but may be weakly connected, holds as soon as a course has at least two prerequisites or develops two or more concepts in the same course.

On the opposite end of the scale, a connected or strongly connected DCCG indicates temporal or logical inconsistencies in the program, since a strongly connected and all relevant connected[2], directed graph must contain at least one cycle. Hence, connected or strongly connected DCCGs are not desirable and hint on the need to further investigate the curriculum structure. See Section III-C below for more details.

2) Minimal cut: In relation to the concept of connectivity, a natural question that arises is to understand which

[2]The only possible, connected DCCG without a cycle is a simple line graph indicating a rather trivial curriculum consisting of a sequence of courses connected by exactly one concept in between, which is developed by the previous course and serves as a prerequisite of the following course. This case can easily be outruled by plotting the DCCG.

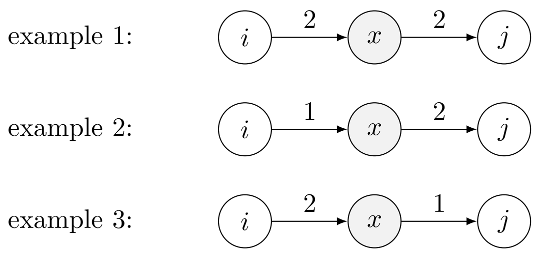

Fig. 3: Examples of basic DCCGs, where i and j refer to courses, and x refers to a concept.

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

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

3) Network flow: The weight of an edge in a network can be interpreted as its capacity to carry a physical flow, e.g., water. Interpreting every edge of a network as such a capacity, it is natural to ask how many units of flow can be transported from one part of the network to another. Defining thus at least one source node and at least one sink node, it is possible to compute the maximal admissible flow that can be carried between these nodes through well-known algorithms, e.g., [7]. Under this maximal flow situation it is possible that some edge carries less flow than their maximum admittable ones; e.g., in example 2 in Figure 3, the maximum flow between nodes i and j is 1, and the edge between x and j is not exploited at its maximum capacity.

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

To give to network flows a pedagogical interpretation under our DCCG framework, we can interpret the weights 0, 1 and 2 in the CCG as capacities that describe two specific phenomena. Referring again to Figure 3, possible interpretations are:

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

A natural question is then about how analysing the maximal network flow associated to a DCCG can then be helpful to understand the structure of a program, its shortcomings, bottlenecks and redundancies.

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

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

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

Assume then to have added to an existing DCCG these fictitious source and sink nodes, their corresponding edges, and to have computed the maximal and the residual flows of the network. Some interesting cases may then arise: First, consider example 1 in Figure 3, where both edges (the one from course i to concept x, and the one from concept x to course j) have the same capacity. Here the maximal flow is such that everything entering in concept x can also flow to course j. This can be interpreted as a well aligned structure, where students are enabled in course i to build up an appropriate knowledge about x, and then use it in course j.

In contrast, examples 2 and 3 in Figure 3 might both reveal some mismatch in the program. In example 2, indeed, the overall maximal flow from i to j is 1, since concept x is only taught with weight 1 in course i even though it is required with weight 2 by the following course j. Hence the residual flow of the edge from x to j is 1, and a plausible interpretation of this is that students may not

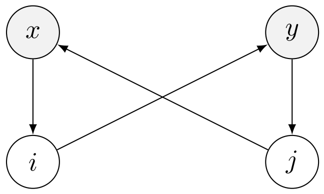

Fig. 4: Example of a cycle in a DCCGs, where i and j refer to courses, while x and y refer to concepts. The weights of the edges are omitted for simplicity.

be as well prepared for taking course j as expected by the teacher of this course. In example 3, it is instead the link from i to x that has under the max-flow regime a residual capacity 1. A plausible interpretation of this is that not all the effort that is put during course i into teaching x is required for the following-up course j. In a sense, reducing this potentially unnecessary effort during course i could free resources that could better be allocated otherwise.

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

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

C. Existence of cycles

Another pedagogically interesting analysis of a DCCG is related to detecting cycles in the graph. For example, consider the situation in Figure 4: here course i has concept x as a required knowledge, and prepares students to course j by teaching concept y. The situation is though symmetric, since j has concept y as a required knowledge, and prepares students to course i by teaching concept x. Thus as soon as i and j are not taught in the same learning period, students will not be prepared to take the first of the courses being taught (something that theoretically will also affect their understanding, eventually inficiating also the attendance to the second one). And even if i and j are instead taught in the same learning period, then this logical fallacy can be resolved only through a great care by the teachers of i and j in designing their own courses so that the understanding and usage of concepts x and y happens in parallel and simultaneously.

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

IV. Results

We gathered data for Electrical Engineering and Computer and Information Engineering at UU, Sweden, and Engineering Physics at Luleå University of Technology (LTU), Sweden, by asking teachers to allocate weights of the scale $\{0, 1, 2\}$ for the concepts in their course in the program according to the methods described in Sections II-A and II-C. The data were then analysed by computing the different indices described in Section III.

The results for the centrality indices for the courses for the two Electrical Engineering and the Engineering Physics programs are visualised in Figure 5 (for LTU) and Figure 6 (for UU). As discussed above, we noticed large variations between how many accumulated weights teachers assigned to a course, which suggests different interpretations on the scale $\{0, 1, 2\}$. Hence, the degree centrality for the courses for the Engineering physics program at LTU reflect the accumulated weights assigned for each course. This highly correlates with the teachers' interpretation of the scale and eagerness to contribute to the project. Further, the pagerank and eigencentrality mostly follow this trend and do not offer additional insights. Lastly, the closeness index is generally high for all courses (with some drops for courses with very low degree) whereas the betweenness is low for most courses with some exceptions for courses with high degrees. In fact, interviewing the head of the corresponding program board, showed that these data do not reflect the perceived importance of the courses in the program but rather the teachers' understanding of the scale.

In order to compensate for this effect, we normalized the weights in the CCM for the Electrical engineering program at UU such that all weights for a course accumulate to one. Hence, the degree centrality, shown in Figure 6 is equal to one for all courses and no information can be extracted for this index. Further, as expected, the betweenness index is low for almost all courses with the exception of some courses, that include a large number of concepts and hence connect many nodes in the graph. For the collected data, the eigenvalue index and closeness index offer insights into the program structure. Indeed, the courses with high eigenvalue and closeness indices were clearly indicated as central courses in the programs by the program board.

Figures 7 and 8 show the centrality indices for all concepts in the CCMs for LTU and UU, respectively. The concepts are ordered by their degree index and are calculated using the original data (i.e., not normalized) for LTU and the normalized data for UU. Note that normalising the degree for each course node to one, does not imply a normalisation of the concept nodes. The degree centrality and the pagerank centrality, which is very similar to the degree, do appear to provide an overall reflection of how often a concept is taken up during the program in both the normalized and the unnormalized case. As expected, the betweenness index is low for almost all concepts. However, for the normalized case (UU) the concepts with high betweenness seem to indicate concepts that are taken up by many different courses. Further, for the normalized CCM, in cases where the degree centrality is low for the same concept, it refers to a concept that is taken up often but also often weighted with a 1 (compared with important concepts receiving weight 2). The opposite seems to be true for closeness in the normalized CCM, which is high for almost all concepts in the program. However, some concepts have a low closeness despite a comparatively high degree. These are concepts that play a relatively strong role in the program but are only taken up in some courses in a rather intensive fashion. Note that such insights cannot be extracted for the unnormalized CCM for the program at LTU.

The data collected for Computer and Information Engineering at UU, instead, comprise information on whether the various given concepts shall be considered a prerequisite or a teaching outcome for the courses. Part of the corresponding DCCG is shown in Figure 9. Analysing the DCCG reveals several interesting insights. First of all, analysing the max flow of the graph allows understanding mismatches in effort in the sense of spending much more time or effort in teaching a concept, that is only required in few or no courses with low weights, or vice versa. For instance, consider "hashtables", which is taught in three courses, 1DL201 (Program Design & Data Structures), 1DT093 (Computer Architectures) and 1DL221 (Objects Oriented Programming) with accumulated weight 4, but only required in one single course, 1DT096 (Operating Systems), with weight 1. In contrast, the concept "induction" is only taught in one early course, 1DT051 (Introduction to Information Technology), with weight 1 but required for three following courses with various weights and further considered an intended learning output of the program. Both cases can be found by analysing the redundancies in the DCCG under max flow. (Links with maximal flow in Figure 9 are shown in green and links with redundancies are shown in blue.)

Further, analysing cycles in the graph allowed extracting mismatches in time, i.e., between courses, where concepts (e.g., "lists" and "arrays") are assumed prior knowledge, or at least a common understanding, and are hence required in an early course (e.g., 1DT093) but are taught in a later course (e.g., 1DT051). These cases are marked as red arrows in Figure 9. Either, teaching these concepts in the later courses is redundant since the teacher of the first course correctly assumed that students would know these concepts from earlier education as for instance high school, or students are not sufficiently prepared for the early course due to a lack of knowledge. It may also be possible that the courses require or teach the concept on different levels in Bloom's taxonomy. In this case, no mismatch might be present. However, in order to understand those issues and to avoid mismatches, teachers need to collaborate and/or exchange more detailed information. In any case, the analysis of the corresponding DCCG allows identification of which aspects need to be discussed or maybe even changed in the course and program organisation.

In summary, these results indicate that relevant information can be extracted from the CCG or DCCG by considering centrality indices of concepts and courses given adequate normalisation of the weights or analysing the connectivity and flow in the graph.

V. Conclusions

In this paper, we proposed a method to analyse quantitative data about which concepts are relevant for which courses in a university program (provided by the corresponding teachers) and the connections between them. We showed how this information can be described by a course-concept matrix (CCM) and a corresponding course-concept graph (CCG). Further, by analysing the centrality indices of the involved courses and concepts for two programs at UU and LTU in Sweden, relevant information could be extracted. It appears that the results are better aligned with the program boards' perceptions and insights if the weights in the CCM are weighted. Further, categorising concepts as required knowledge for a course or taught in a course allows building a related DCCG. Its analysis reveals mismatches and redundancies in the program.

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

In this first attempt to systemize concepts of an engineering program and a tool to understand the interconnections between them, we can conclude that concepts are probably not enough to describe the learned content from courses and the whole program. Motivated by [16] one should rather include facts and procedural knowledge along cite conceptual knowledge. Further, instead of using the simple scale from 0 to 2, using the levels ranging from "remember" to "create" from Bloom's revised taxonomy, [2], is expected to further improve the analysis of the university program. However, the proposed graph theory tools have to be carefully revised in order to obtain meaningful results when using the taxonomy.

Also, suitable methods should be developed to visualise and display the graph structure and highlight the obtained results such as flows, cycles and discrepancies in a suitable way. Even though our preliminary experience in communicating our method to students, boards, administrators and fellow teachers revealed great interest and readiness to adopt the new method, its success will depend on whether the information can be presented in a useful and understandable way for all stakeholders. We believe that such a suitable visualisation would also help to realise some of the goals outlined in [11] such as

- informing learners of what they can expect to achieve from a program, so they can organise their time and efforts;
- communicating curriculum/program goals in a meaningful way to a broader community;
- helping to determine the extent to which learning has been accomplished;
- guiding curriculum committees (within resource constraints) to determine program(s) of study and course offerings;
- guiding instructors when they are designing course objectives, content, delivery and assessment strategies.

However, how to reach those goals with the help of the here proposed tool and its extensions and further developments needs to be investigated in the future. Another important direction for future development is to include actors outside the academia, such as the industry, both by using the (graphical) tool to allow them to get a much clearer view on content, (program) goals and the intended progress for each program as well as considering and combining their demands with the program learning outcomes.

References

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

[2] Lorin W Anderson, David R Krathwohl, Peter W Airasian, Kathleen A Cruikshank, Richard E Mayer, Paul R Pintrich, James Raths, and Merlin C Wittrock. A taxonomy for learning, teaching, and assessing: A revision of bloom's taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman*, 2001.

[3] M Battersby. Outcomes-based education: A college faculty perspective. *Learn Quarterly*, 1:6–11, 1997.

[4] F Michael Connelly, Ming Fang He, and JoAnn Phillion. *The Sage handbook of curriculum and instruction.* Sage, 2008.

[5] Edward F Crawley, Johan Malmqvist, Sören Östlund, Doris R Brodeur, and Kristina Edström. The CDIO approach. In *Rethinking engineering education*, pages 11–45. Springer, 2014.

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

[7] Reinhard Diestel. *Graph Theory.* Springer-Verlag Heidelberg, New York, 2005.

[8] European Commission. The Bologna Process and the European Higher Education Area. https://ec.europa.eu/education/policies/higher-education/bologna-process-and-european-higher-education-area_en.

[9] Eva Fjällström, Steffi Knorn, Kjell Staffas, and Damiano Varagnolo. Developing concept inventory tests for electrical engineering: extractable information, early results, and learned lessons. In *Proceedings of the UK Automatic Control Conference*, 2018.

[10] Bernard SM Gatawa. *The politics of the school curriculum: An introduction.* College press, 1990.

[11] Harry Hubball and Helen Burt. An integrated approach to developing and implementing learning-centred curricula. *International Journal for Academic Development*, 9(1):51–65, 2004.

[12] Jay M Lightfoot. A graph-theoretic approach to improved curriculum structure and assessment placement. *Communications of the IIMA*, 10(2):5, 2010.

[13] Jaime A. Pavlich-Mariscal, Mariela Curiel, and German Chavarro. CDIO curriculum design for computing: a graph-based approach. In *Proceedings of the 15th International CDIO Conference*, 2019.

[14] Raita Rollande. *The Research and Implementation of Personalized Study Planning as a Component of Pedagogical Module.* PhD thesis, Riga Technical University, 2015.

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

[16] Kjell Staffas. *Developing requisite motivation in engineering studies: A study on a master and bachelor program in electronic engineering at Uppsala University.* PhD thesis, Aalborg Universitet, 2017.

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

[18] Gabriela C Weaver, Wilella D Burgess, Amy L Childress, and Linda Slakey. *Transforming institutions: undergraduate STEM education for the 21st century.* Purdue University Press, 2015.

[19] Jon Wiles. *Leading curriculum development.* Corwin Press, 2008.

[20] Jon Wiles, Joseph Bondi, and Hua Guo. *Curriculum development: A guide to practice.* Merrill Publishing Company, 1989.

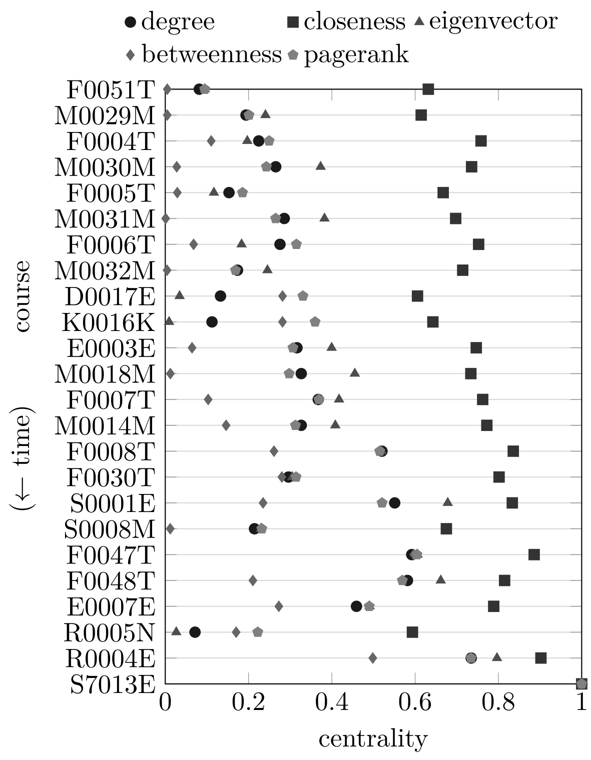

Fig. 5: Measured courses centrality indexes for Engineering physics at LTU, Luleå, Sweden.

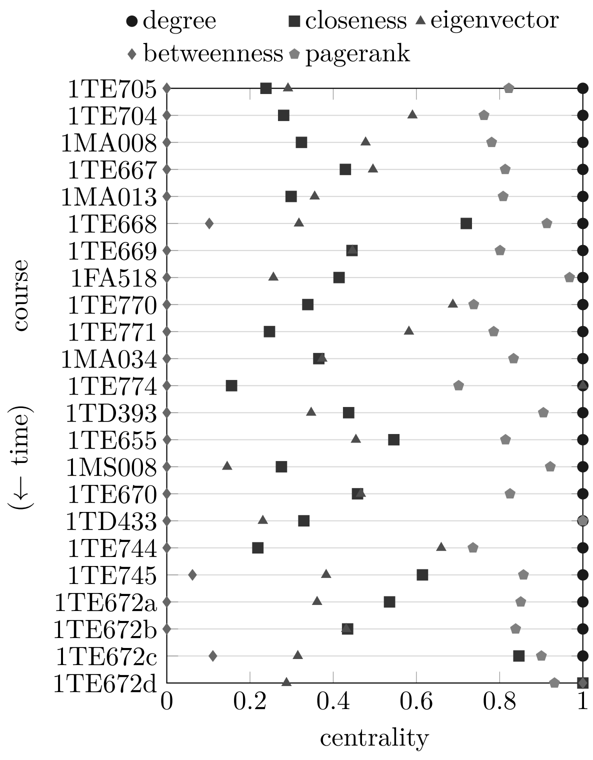

Fig. 6: Measured courses centrality indexes for Electrical Engineering at UU, Uppsala, Sweden.

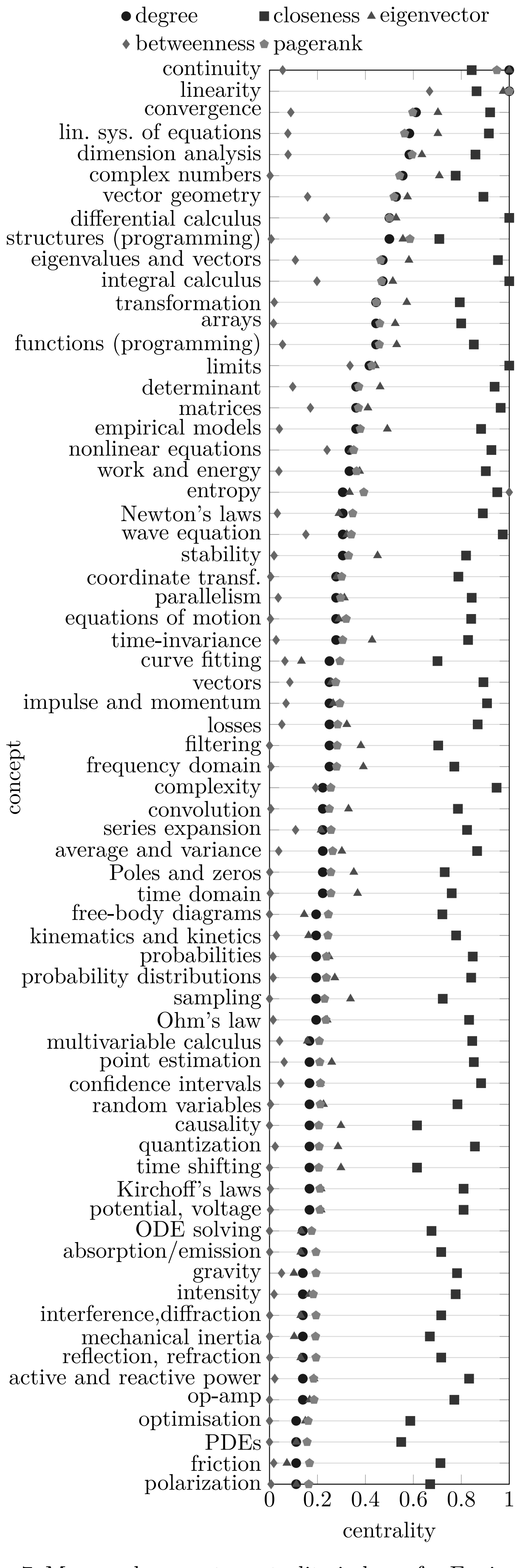

Fig. 7: Measured concepts centrality indexes for Engineering physics at LTU, Luleå, Sweden.

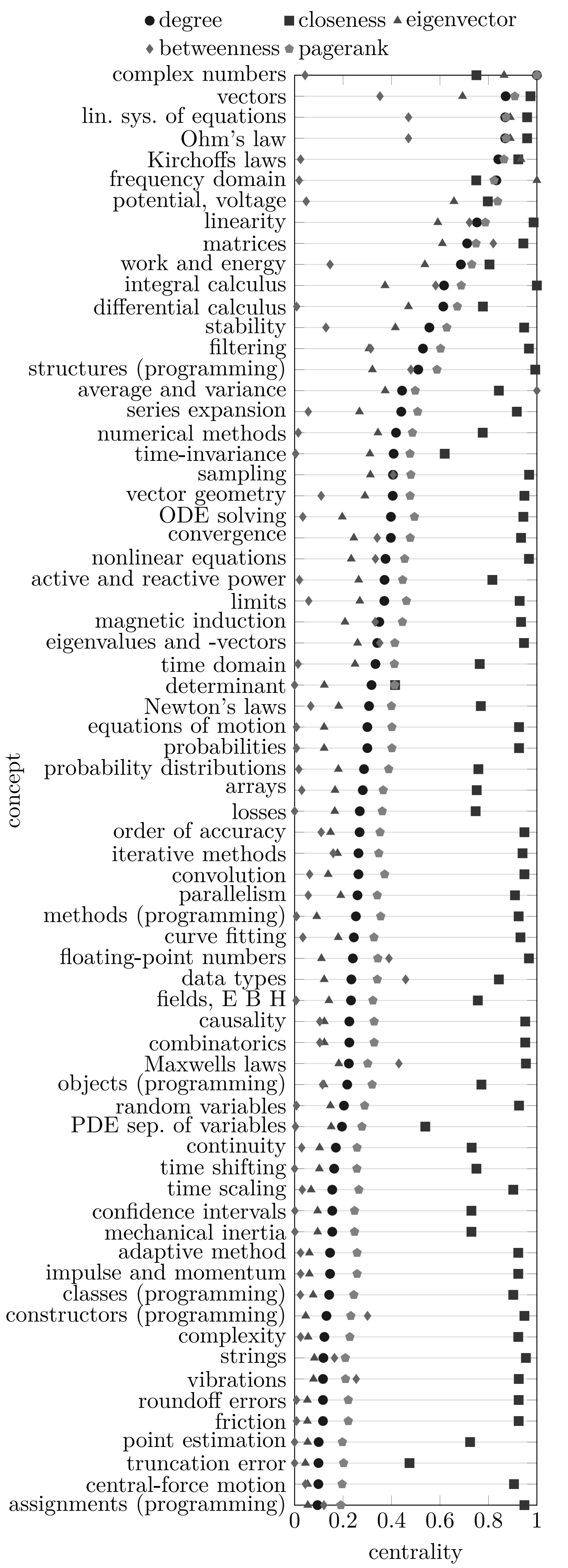

Fig. 8: Measured concepts centrality indexes for Electrical Engineering at UU, Uppsala, Sweden.

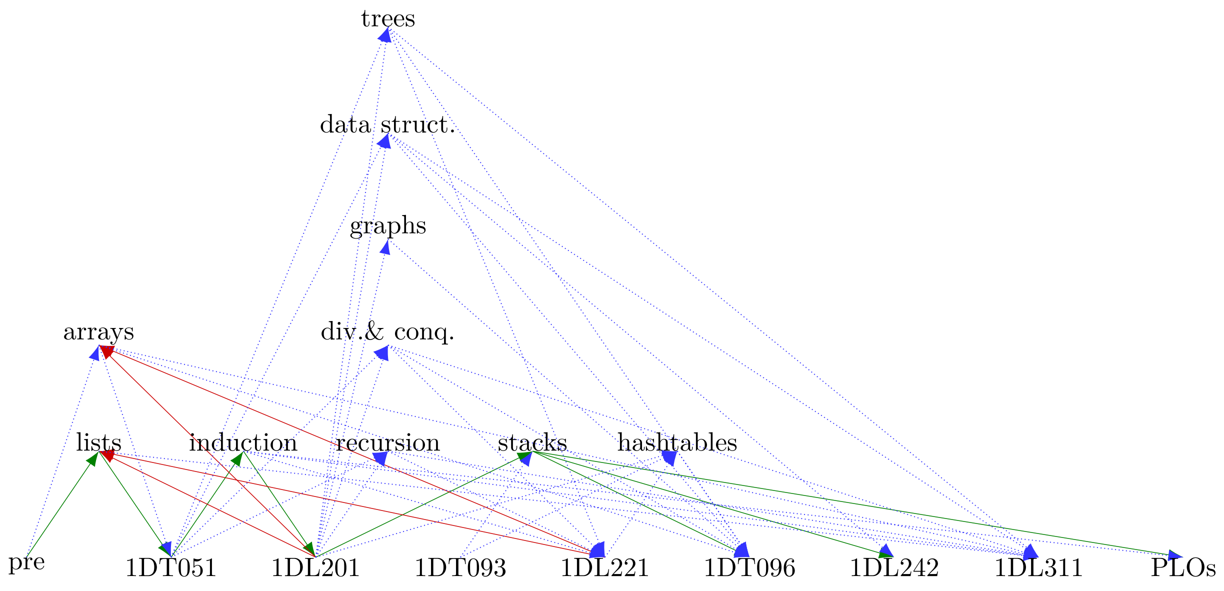

Fig. 9: Part of the DCCG for Computer and Information Engineering at UU showing temporal mismatches as red arrows, unused/redundant connections as blue arrows and remaining (well matched) connections are green arrows. (Note that the max-flow analysis involved all concepts but only 10 are shown here.)