CS-Materials: A System for Classifying and Analyzing Pedagogical Materials to Improve Adoption of Parallel and Distributed Computing Topics in Early CS Courses

Alec Goncharow, Matthew Mcquaigue, Erik Saule, Kalpathi Subramanian

Dept. of Computer Science
The University of N. Carolina at Charlotte
Charlotte, NC 28223

Paula Goolkasian

Dept. of Psychological Science
The University of N. Carolina at Charlotte
Charlotte, NC 28223

Jamie Payton

Dept. of Computer Science
Temple University
Philadelphia, PA 19122

Abstract

The NSF/IEEE-TCPP Parallel and Distributed Computing curriculum guidelines released in 2012 (PDC12) represents an effort to bring more parallel computing concepts into early computer science courses. To date, it has been moderately successful, with the inclusion of some PDC topics in the ACM/IEEE Computer Science curriculum guidelines in 2013 (CS13) and mentions of PDC topics in the Computing Curricula 2020. Additionally, some universities in the U.S. and around the world have started to cover some of these topics in early CS courses. Lack of knowledge of or training in PDC topics among instructors, along with the need to align early CS course content with prescribed learning objectives in the curricula, are often cited as hurdles for adoption in early CS courses. There have been attempts at bringing PDC materials, such as textbook

[1]{agoncha1,mmcquaig,esaule,krs}@uncc.edu
[2]pagoolka@uncc.edu
[3]payton@temple.edu

chapters, lecture slides, assignments, and demos to assist instructors of early CS classes. However, the effort required on the part of the instructor to figure out what is relevant to a particular class can be daunting.

In this work, we contend that simultaneously classifying pedagogical materials against the CS13 and the PDC12 curriculum guidelines can address some of the challenges faced by instructors and can promote broader adoption of PDC materials in early CS courses. We present *CS Materials*, a system that can be used to categorize pedagogical materials according to well-known and established curricular guidelines. We show that *CS Materials* can be leveraged 1) by instructors of early CS courses to find materials that are similar to the one that they use but that also cover PDC topics, and 2) by instructors to check the coverage of topics (and gaps) in a course, and 3) by PDC experts to identify topics for which PDC instructional materials do not exist or are insufficient in order to inform development of additional PDC curricular materials.

---

1. Introduction

As Dennard scaling came to an end around 2005 and as the ubquity of Internet systems continued to increase, interest in the area of Parallel and Distributed Computing (PDC) has grown substantially. However, PDC topics still have not reached the classroom; few students enrolled in computer science degree programs programs are exposed to these important concepts. To provide guidance that could help to drive the development of courses that cover PDF concepts, the NSF/IEEE-TCPP curriculum guidelines were developed and published in 2012 [1] [4]. Further, the importance of PDC in Computer Science curriculum was recognized and a joint ACM/IEEE group integrated PDC topics in their 2013 Computer Science curriculum guidelines [2] as a dedicated area, as well as spread them in multiple places in the guidelines. In parallel, Computing

[4] As topics in PDC and computing education itself have matured, a new iteration of thes NSF/IEEE-TCPP guidelines has been developed, with a beta version released in 2020 and a finalized version expected in 2021

Curricula 2005(CC 2005) [3] and the newer Computing Curriculua 2020(CC 2020) [4, 5, 6] guidelines focus on computing guidelines at a higher level, in terms of distinctions between computing programs, international and cultural factors in the programs, and variations in discipline specific computing programs, that can be useful to inform academia, industry and governments. It is to be noted that these standards emphasize the importance of parallel and distributed computing in today's curricula and the recent CC2020 guidelines include PDC concepts as key concepts for all the various computing programs it surveys.

While at a national level there is an understanding that PDC topics are important, the practical integration of these topics in curricula has been slow. Rather than proposing to add a PDC course in curriculum, a more promising strategy for widespread adoption and broad student exposure is to integrate PDC topics across the undergraduate curriculum, from early CS courses, such as introductory programming, data structures and algorithms, to more advanced courses, such as operating systems, and computer architecture. Although this may be a more viable approach, multiple strategies to help with the adoption of PDC integrated across existing courses and curricula have been deployed with moderate success. Workshops to train instructors are effective at adjusting some courses, but the strategy is slow and not scalable. Books to explain how to teach these topics provide some materials and guidance [7, 8], but are not timely, tend to be very specific, and need to be well publicized to reach a broad audience.

From an early CS course instructor's perspective, the questions that arise include "What topics should I adopt in my class?", and "How do I adopt them in an already packed class?" A set of well developed learning materials (assignments, videos, textbooks, course descriptions, and so on) is a starting point to such questions, and, indeed, PDC materials have been developed and made available online [9, 10, 11]; however, this leads to questions such as "How do I find them?" and "Which ones are relevant to my course and learning objectives?".

In this work, we propose a way to address this issue by taking advantage of

two nationally accepted curriculum standards by classifying pedagogical materials, such as assignments, lecture slides, exams, video lectures, book chapters, etc. against their well accepted content ontologies. We classify learning materials against the 2013 ACM/IEEE CS curriculum guidelines [2], and the 2012 NSF/IEEE-TCPP curriculum guidelines for Parallel and Distributed Computing [1]. Assessing materials against accepted curriculum standards lends credibility to the content and learning objectives of the materials, and assists search and comparative analysis between sets of materials, which are essential to improving the broader adoption of PDC materials. The CS Materials system benefits both individual instructors as well as PDC experts looking to contribute to the computing community. CS Materials provides a robust means for instructors to assess their own courses against national guidelines, to ensure alignment on content and learning objectives, to take a deeper look at their own courses, and possibly to make revisions leading to improvements. When populated with sufficient amount of materials, CS Materials will let PDC experts look at existing content on early CS courses, identify the current models for core courses, and find opportunities to bring in equivalent PDC materials to instructors; the latter can be accomplished with robust search and comparative analysis features.

We have developed *CS Materials* [5], a system that permits instructors system to classify their materials against curriculum standards. Currently, the system supports the ACM 2013 and ACM/IEEE TCPP PDC guidelines. The system currently contains a collection of classified materials that include all the Nifty Assignments [12], all the Peachy Parallel assignments [9], all the materials used to teach five courses, including a parallel and distributed computing course [13], two flavors of a sophomore level data structures courses, a junior/senior level courses on software engineering and object oriented programming courses, all of them taught by three of the authors at UNC Charlotte. In addition, four additional courses from instructors external to UNC Charlotte are in the process of being classified and entered into the system, spanning CS1/CS2, and data

[5]System available at: `https://cs-materials.herokuapp.com/`

structures.

A preliminary version of this paper appeared in Edupar 2018 [13], which described an early prototype of our system which was single user and only had basic search and analysis features. This paper presents the production-ready system which has been rewritten for scalability, a multi-user environment with file upload, and support for manipulation sets of materials. The current version of the system is significantly more robust and has been used by external users. The new system supports sophisticated search and similarity matching which were not possible in our prototype system. The new system supports harmonization views to improve the classification effort. A single class was classified in [13] while we have now classified 7 courses in the system which enables us to demonstrate features that cater for both individual instructors as well as PDC experts looking to find gaps in materials targeted towards early CS courses. We present the feedback of two external users of the system, showing a perspective on the system that does not come from the authors.

We showed in [14] how CS Material can be used to design a course (or, any course), by studying its coverage of topics, the alignment of material between different component. This manuscript, on the other hand, focuses on how the system can be used to improve the adoption of PDC in early CS courses: in particular, it is used to compare Peachy Parallel Assignment and Nifty Assignment which would be useful to Peachy Parallel Assignment designers; it shows how the system can be used to build detailed models of a well understood class (such as "Data Structures"), which would be useful to PDC experts; it also features an ontology-aware search which would be useful for instructors shopping for PDC content.

In this paper, we show the new features of the system enable a much deeper analysis of the field and are practically useful to both instructors of courses and PDC education experts. We demonstrate how classifying pedagogical material can help improve the adoption of PDC topics in early CS courses by tackling three different problems: 1) help PDC experts identify topics for which pedagogical material does not exist and that should be developed, 2) help instructors

of early CS courses to find materials that are similar to the ones they use but that also cover PDC topics, and, 3) help instructors to check the topics that their course currently covers and the ones it does not cover (and maybe should or could).

2. Related Work

2.1. Learning Material Repositories

Nifty Assignments. The Nifty assignments repository [12] is a set of assignments that have been collected since 1999 (over 100 assignments) through an annual competition, as part of the ACM SIGCSE conference. Selected assignments are presented at the conference and archived. The selection is primarily based on engagement, adoptability and scalability, and usually targeted at early courses (CS0, CS1, CS2). Nifty assignments now include metadata (topics, difficulty, strengths/weaknesses, dependencies, variants). The Nifty assignments are used as part of the initial assignment set by CS Materials, as they represent a classical (non-PDC) learning materials for early CS courses. Some specific CS subcommunities have created their own assignment repository patterned after Nifty Assignments such as Model AI [15] for Artificial Intelligence and Groovy Graphics [16] for computer graphics.

Peachy Parallel Assignments. The Peachy Parallel Assignments [9] are a recent effort of the EduPar and EduHPC [17] workshops to publicize well designed, exciting, and interesting assignments that include some parallel and distributed computing aspects. Peachy assignments focus on adoptability and have been succesfully used in a real classroom. The assignments are peer reviewed and published and presented at EduPar and EduHPC; so far 11 Peachy Parallel Assignments have been presented in 2018 and 9 were published in 2019. The Peachy Parallel Assignments are used as part of the initial set of the CS Materials system as a representative set of what could be gathered from the PDC educational community.

EngageCSEdu. EngageCSEdu [18, 19] is an NCWIT sponsored repository that provides introductory CS course materials, primarily engaging assignments targeted at CS0, CS1, and CS2. The assignments are categorized by engagement practices to improve student inclusiveness, confidence and broadening participation in computing. The repository has over 800 assignments with a competition for excellence [20] and the assignments submitted are subject to an editorial process with peer review.

Data Repositories. The CORGIS data repository [21, 22] is a large collection of tools, datasets and resources that can be used by educators as part of their programming assignments. The datasets range across a large number of disciplines and have been used in introductory courses, such as Computational Thinking [23]. Using real-world datasets can be highly engaging in introductory courses. Real-world applications and dataset have been successfully integrated in Data Structures courses [24, 25]. CS Materials includes the usage of datasets as a dimension of interest for assignments.

CS in Parallel. CS in Parallel [10] is a repository containing a limited set of learning materials that are used to teach parallel computing in various classes. Overall, the repository contains a few documents and organizes them according to the courses they fit into. Organizing materials by courses is difficult since different universities may have very different ways to organize their curriculum and the topics within their curriculum. We believe it is preferable to classify against well accepted topics in order to enable the percolation of the material in early CS courses by empowering instructors to decide which topics fit best in their classes. We will see that CS Materials can be used to build models of courses, removing the uncertainty of which class a material could be useful in.

PDC Unplugged. PDC Unplugged is a repository that contains a collection of unplugged parallel and distributed computing activities [11]. The activities span early CS courses (CS0, CS1 and CS2), sophomore courses (Data Structures and Algorithms) and advanced courses. There are a total of about 130 activities

spanning these courses. These activities provide an easy introduction to PDC materials and can be part of an overall assignment, that begins with unplugged activities and then leading to a more formal programming exercise. PDC Unplugged share a common strategy in that the unplugged activities are associated with the entries of the curriculum guidelines that the activities covers. CS Materials goes further and advocates that the activities should be mapped to all the entries in the curriculum guidelines that are relevant and not only the ones related to PDC.

The CDER Courseware. is a repository of documents related to PDC education. The material can be associated with some entries of the CDER PDC12 guidelines and with courses (which as described before requires submitters to make judgment calls so as to which course a material is appropriate for). Also the courseware contains lots of document that are not intended for class usage such as educational paper, posters, and so on. The documents in CDER courseware are not classified against the ACM/IEEE curriculum guideline. All these fact make it hard for non-PDC experts to identify the materials that are relevant to them. Historically, CDER Courseware is the prototype database that inspired CS Materials; and the authors of this paper hope that CS Materials can one day replace the CDER Courseware.

Other Repositories. Other repositories include those surveyed by Decker et al. [26] that also include learning materials for high school teachers, and detail their barrier to entry/participation [27].

Overall, existing repositories tend to only consider early computing education by focusing on courses such as CS0, CS1, and CS2; do not provide classification against well accepted content ontologies; and tend to focus on assignments rather than all learning materials. In comparison, the CS Materials system aims to include a wide range of computer science topics and to provide a more expansive, fine-grained classification system that allows for greater expressiveness in search queries to finding new, equivalent or engaging materials, assess one's own materials for alignment within a course or to national curriculum standards.

2.2. Curriculum Guidelines/Standards

ACM regularly updates computing curriculum guidelines and the latest Computer Science curriculum is from 2013 [2], jointly sponsored by ACM and the IEEE Computer Society (we will denote this guideline CS13). The CS13 guidelines specify a 'redefined body of knowledge, a result of rethinking the essentials necessary for a Computer Science curriculum'. The guidelines also provide numerous exemplars of actual courses and programs that can be adopted by CS departments. In short, the guidelines divide the body of knowledge into a set of *knowledge areas*; knowledge areas are further divided into into *knowledge units* which contain *topics* and *learning outcomes*. Learning outcomes are classified into three levels, *familiarity*, *usage* and *assessment*. The system we propose adopts the classification proposed by the ACM CS13 curriculum guidelines as a general Computer Science curriculum since it is widely accepted.

The 2012 NSF/IEEE-TCPP curriculum for Parallel Distributed Computing [1] (we will denote PDC12) is an effort to accurately map the PDC topics that are necessary for all students to know. It is divided in four areas: Algorithm, Architecture, Programming, and Cross-Cutting and Advanced topics. Contrary to the CS13 guidelines, the PDC12 curriculum presents learning outcomes only as a description of topics rather than as separate items. The PDC guidelines also associate Bloom levels [28] (such as Know, Comprehend, and Apply) with the topics to clarify the minimum level of understanding a student should have. While the CS13 curriculum groups topics into a core-1 (must cover 100%), core-2 (should cover 80% at least), and elective; the PDC curriculum only exposes two levels: core and elective. The PDC curriculum is currently under revision with a new version coming in 2021 (a beta version was released in late 2020). We used the NSF/IEEE-TCPP PDC12 curriculum in CS Materials as a domain specific curriculum.

It is noteworthy that CDER is currently redefining its curriculum guidelines and that a draft is expected soon. Also, CC2020 [4] is a currently maturing effort to publish recommendation of what the different computing degrees are meant to be. For instance it intends to delineate the differences between Computer

Science, Computer Engineering, Information Technology, and so on. While CC2020 notes that PDC is a key field to be covered by all degrees, the description of the degrees and their recommended content is at a too high level to be directly useful in CS Materials.

Other sub-areas of computing have developed their own standards, such as cyber security [29] and high school CS curriculum [30, 31] which could also be used to provide analyses similar to the one we conduct.

An earlier project that shares some goals with CS Materials is a syllabus repository project of Tungare et al. [32]. The authors of this work built a repository of syllabi of computing courses by crawling the Internet and classified them against the 2001 Computing Curricula standards [33]. They also proposed a syllabus maker and a comparison tool. Syllabi are often fairly high level and are concerned only about the primary topics intended by a course. CS Materials on the other hand looks deeper into courses by looking at the level of learning materials (lecture slides, exams, videos). While it is possible to enter a syllabus in CS Materials, CS Materials facilitates much deeper analyses and search features, and takes more of data-driven approach to its analysis based on the *actual content of the course and not on what is written in a syllabus*.

3. The CS Materials system

3.1. Needs

Instructors are often looking for inspiration for new lectures, problem sets, and exercises that must align with computing education curriculum standards (e.g., ACM Curriculum Guidelines, ABET standards) and that adequately address the learning objectives of their courses. However, traditional search tools are not helpful in finding equivalent assignments or learning materials that precisely match their own course and learning objectives. The lack of centralization of materials certainly presents a problem, but the most significant limitation is the lack of meaningful, searchable features that that can make a material useful for a given computer science course. Labeling by course title, as is done for

most existing collections of course assignments and materials [10, 12], is too simplistic a description as course content can vary widely across institutions; for instance, not everyone agrees on what CS1 should cover. Flavors of CS1 can vary in terms of course period (quarter, semester), the choice of programming language can result in different specifications of assignments. Moreover, instructors are often looking for new ways to create assignments that promote student engagement and are relevant to a diverse student population. They are also looking for materials that can substitute what they are currently using, without straying too far in terms of their learning objectives. Some instructors may use a particular pedagogical approach, instructional technique, or running theme for assignments, for instance, media computation, use of social media data, etc.

3.2. Our approach

The CS Materials system addresses the need for more meaningful searches of computer science course materials. It enables finding materials that are defined by concrete topics, learning objectives and/or outcomes. The CS Materials system pairs learning materials with properly curated metadata to create a more fine-grained and structured representation, to facilitate sophisticated search. The system uses classic material descriptors, such as course level, programming language, and datasets. More importantly, CS Materials also *relates materials to curriculum ontologies*. Currently, the system can classify materials against the ACM Computer Science 2013 curriculum guidelines [2] and the NSF/IEEE-TCPP Parallel and Distributed Computing 2012 curriculum [1] to extract more meaningful, discipline-specific, fine-grained features to describe each material. Specifically, each material will be associated with the topics covered by the material and the learning outcomes it fulfills. Note that the system's design supports the inclusion of additional standards and guidelines, such as the ones detailed in Section 2.2, making it highly flexible to support many custom environments, for instance, a curriculum at a particular institution.

Mapping material to curriculum guidelines and other descriptive features

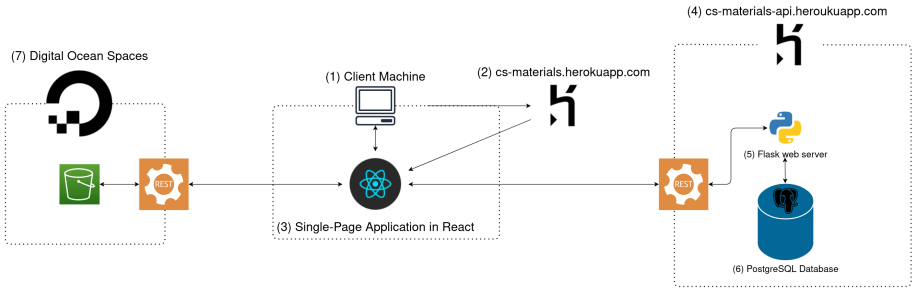


Figure 1: Infrastructure design of CS Materials

opens up several opportunities for new search and analysis functionalities. For instance, one can explicitly filter against a group of features that is of interest to an instructor looking for a specific material, or look for similarities to an existing material, and perhaps, create variants of an existing material. It enables one to ask questions pertinent to a material or to a course, or to understand how a topic or a learning outcome is typically covered.

We rely on a crowdsourced model to address the need for curation. With such an approach, *instructors* can upload their own material in the system and a number of *editors* can review the uploaded materials. An editor has experience or credentials demonstrating knowledge of the standards used by the system, and can appropriately edit or fix classification issues with a submitted material. We envision that eventually less knowledgeable *users* will be able to suggest changes to the metadata which can then be verified by an *editor*.

3.3. Implementation

CS-Materials is a modular system, utilizing two web services hosted on Heroku. While the system is currently closed source, we will eventually open source it. The system design is depicted in Figure 1. One service is used to distribute the user interface as a React single-page application implemented in TypeScript. The application supports dynamic queries thanks to the asynchronous communication with the RESTful API. Additional interactivity and visualizations are provided by the D3 JavaScript library [34].

The second service is a Flask webserver which provides RESTful API to a PostgreSQL database and DigitalOcean Spaces for file storage. In the database, each material is associated with a type, title, authors, URL and description. Materials are able to map to "tags" via an intermediate many-to-many table. Each tag is associated with a title, a type, and an if applicable a key to an entry in the "Ontology" table. The "Ontology" table encompasses our high level Curriculum Guidelines representations, with the shared attribute of each entry having some mapping to a parent node in its Curriculum Guideline if it is not a root node. Our current implementation uses the ACM CS13 Curriculum Guidelines [2] to classify material and the NSF/IEEE-TCPP PDC12 Curriculum Guidelines [1].

In order to handle access control and authentication, the database also includes a "User" table which keeps track of user emails for verification and password resets. User passwords are stored as a one way hashed and salted key, utilizing the cryptographically secure scrypt key derivation function for generation and verification. Materials and users do not store the relationship between one another on their respective tables, instead these relationships are tracked on another many-to-many table, similar to how the material and tag relationships are tracked. The RESTful API and user interface utilize JSON Web Tokens for authentication of user's identity.

The final relationship is between materials and other materials, where also we utilize a many-to-many table for this relationship tracking. This allows us to represent collections of materials in a way that does not require the curator of a collection to necessarily have write access to the materials they would like to include.

The user interface provides a form for entering the free form information about materials, such as the title, description and an upstream URL for reference. All the other fields are instances of tags. Since every tag exists as its own entry in the database, we are able to provide autocompletion for each tag field, utilizing existing tags of the same type to generate a dictionary. The intent of this feature is two-fold, it simplifies the inputing tags for users while also try-

ing to keep all materials utilizing the same tag entry in the database for easier searching and comparing. "Ontology" tags operate similarly to the free form tags but are instead laid out in a tree structure, using the same structure provided by the curriculum guidelines. The current system allows users to create materials which only they have write access to. The database design supports an arbitrary number of relationships between materials and users, with the intent to support inviting other users to review and edit materials in a collaborative manner.

Along with classifying the material, the system supports file storage on a per material basis. The current system is implemented using DigitalOcean Spaces, which is an Amazon Web Services S3-compatible object storage service. When a material is displayed in the user interface, the system is able to do a lookup in the file storage service based on the material's primary key, getting links for each file mapped to that material. Similarly, when uploading a file, the system requests a presigned URL from the RESTful API which allows the client to upload their file directly to DigitalOcean. This design choice was made to minimize redundant file transfers.

4. System Features

4.1. Inputting materials

4.1.1. Inputting a single material

A common task for instructors is adding a new learning material (e.g., project, lecture slides) to the system and classifying it according to a given set of topics, learning objectives, or curriculum guidelines. A web form guides the user to share basic information (e.g., title, authors, description) used to build the item's metadata. An example is shown in Figure 2(a) for mapping part of a Nifty assignment to the ACM 13 classification. The leaf nodes in the tree list in Figure 2(b) represent topics or learning outcomes in the curriculum guidelines/standards, and can be selected to indicate that the particular topic is covered by the assignment; a similar hierarchy is available for selecting learning

outcomes. After the selection is completed, the newly added material displays the selected mappings, as seen in the bottom of Figure 2(a).

One can also upload files of the materials into the system rather than just giving external links. We have found that other systems that reference pedagogical materials eventually run into dead-link problems, due to the documents being no longer available. Having the materials as part of the system is hence, preferable.

4.1.2. Inputting a collection of materials

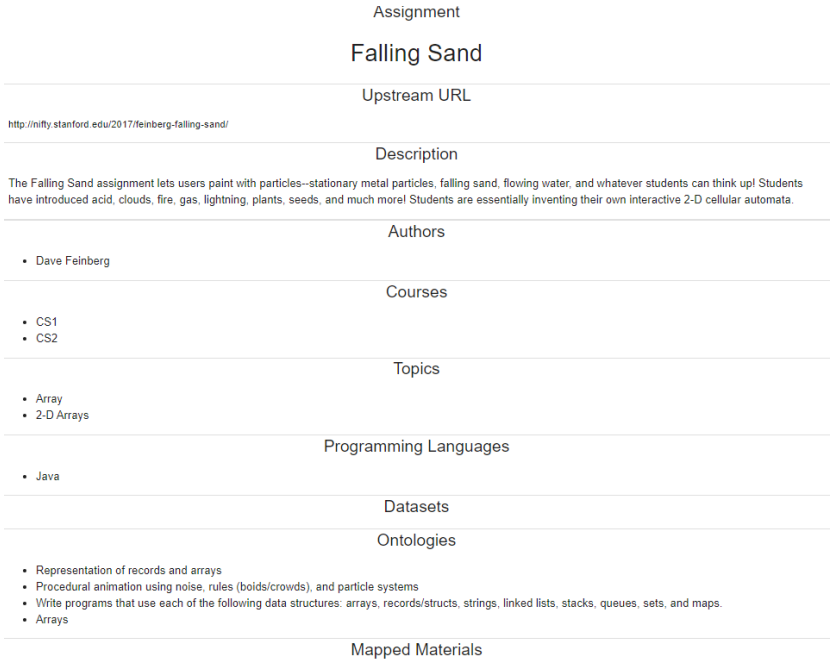
The system also allows the user to define collections of materials. This is a useful feature, in that (1) it lets instructors structure their materials into logical groupings (lecture slides, assignments, quizzes), and, (2) facilitate analyses between collections, such as comparing the lecture slides of two flavors of the same course, and ensuring they are well aligned to learning objectives.

4.1.3. Editing the classification of multiple materials at once

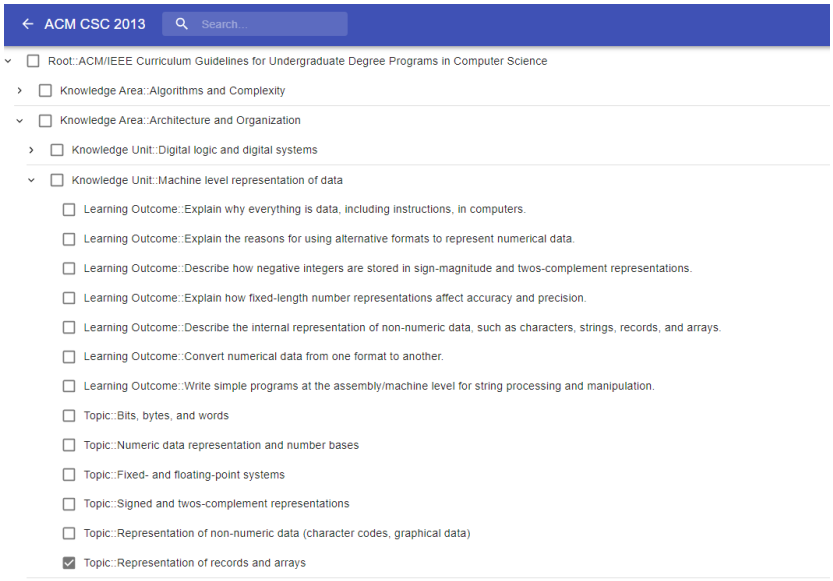
The classification is complex, requiring the user to go through the procedure a few times to become familiarity with how the topics and learning outcomes are organized. For instance, any typical DS course maps to topics from at least three Knowledge Areas: *Software Development Fundamentals, Algorithms and Complexity, Discrete Structures*. After having classified a few materials, one typically realizes that a second pass will be necessary.

CS Materials provide an harmonization view that enables users to visually edit the curriculum mapping of multiple materials at once. (See Figure 3 for reference.) The harmonization view is a table that displays the materials as columns and the classification tags as rows. The table shows whether a material maps to a curriculum entry by coloring the cell red; it is white otherwise. One can directly click on a cell to add a mapping (turning it blue) or to remove a mapping (turning it grey).

To help the user make sense of the materials and their curriculum mapping, the harmonization view permutes the rows and columns of the matrix to



(a) Entering metadata of material into a web form



(b) classifying the material to the ACM CS13 curriculum guidelines

Figure 2: Adding and Classifying a Nifty Assignment

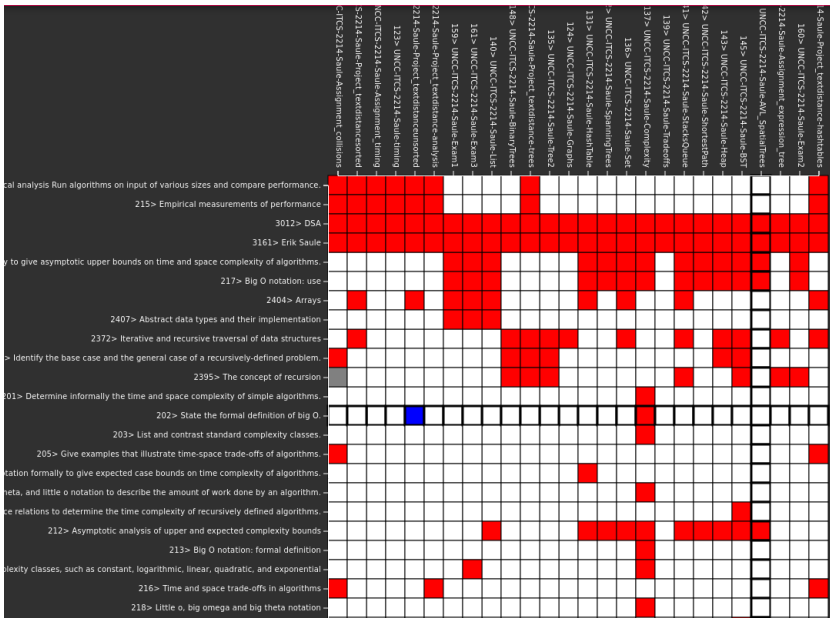


Figure 3: Harmonization View: Verifying and editing the classification of a Data Structure class using the harmonization view

highlight recurring patterns. In practice, CS Materials solves a bi-clustering problem, a type of problem commonly used in bioinformatics to identify correlated up-regulation of genes in a population of subjects [35].

4.2. Visualizing how Materials Map to Curriculum Guidelines

4.2.1. Coverage Views

Users can visualize the coverage of a course (as defined by its collection of associated materials in the CS Materials system) in terms of topics or objectives by viewing a *hit-tree*. The hit-tree is a tree representation with items associated with the course are highlighted in a subset of the ACM classification tree. Example views can be seen in Figure 4: nodes in orange are the selected topics or learning outcomes, those in blue represent nodes that are on the path to a selected node and the remaining gray nodes provide context to the selections. The selected nodes are sized according to the number of times the topic or outcome was selected throughout the course.

The tree is radially laid out by first identifying the level of the tree that has the most nodes called the reference level. These nodes will all be at the same level and be uniformly spaced. Then all the other nodes allocated an angle that is proportional to the number of nodes in the reference level that are upstream (or downstream) from the node considered. To make better usage of space, the nodes are laid out in multiple layers (precisely, 3) for each level of the tree. Finally, the radial distance from one level of the tree to the next is computed to avoid node overlap while maintaining a good density of the visualization.

4.2.2. Difference Views

Users can also explore alignment of learning materials within or between courses or modules; for instance, how well does the material align with the module or course objectives, between different materials within a course (assessments vs. lecture slides), and even equivalent materials between two sections of a course. For this task, we need a way to compare different sets of materials. We first create an alignment tree for the two sets of materials, S_1 and S_2, which is a subtree of the classification (CS13 or PDC12); a node in the tree is in the alignment tree if a mapped item appears in any of the materials of S_1 or S_2. Visually, the nodes of the tree are sized based on the number of materials that are mapped against that item. The nodes are colored on a diverging color scale, where the color represents the difference between the relative number of items in S_1 that are mapped against that item and the same item in S_2. Figure 6 presents such an alignment tree.

4.2.3. Focusing on Entries of Interest

The Radial view can also be configured to only display particular entries of the classification. The visualization takes a set of classification entries as a parameter. Only these entries and their parents, all the way to the root of the classification tree, are displayed.

The styling of the nodes of the tree are left the same. That is to say that a displayed entry that is not covered by any material will be grayed out, and all

classification entries will be sized based on the number of materials that map to that entry.

4.3. Searching for Materials

One main advantage in classifying pedagogical materials against curriculum guidelines is that we gain rich meta data that describe pedagogical materials. These can be used to perform searches for materials.

The original system [13] used Jaccard coefficients [36] as a metric of similarity. We found that using the Jaccard coefficient to estimate similarity did not yield satisfactory search results because it uses a very narrow definition of similarity where two classification entries are either the same or they are completely different. We now present a more sophisticated similarity measure used by the current version of CS Materials that leverages the ontology to relate classification entries that are not exactly the same.

4.3.1. Ontology-Aware Similarity

To leverage the underlying ontology of the classification entries, one needs to be able to relate different entries in the classification to one another. The textual description of classification entries may be of help, but making sense of it could be difficult because of homonyms and synonyms: for instance, the heap data structure and the memory heap are completely different concepts. Similarly, while graphs are also referred to as networks, they are very different from computer networks.

We use the hierarchical structure of the classification tree to estimate the similarity between two classification entries. The similarity will entirely be defined based on the location of the entries in the classification tree. For any two entries t_1 and t_2, if they are the same then $sim(t_1, t_2) = 1$. Otherwise, we will identify the most recent common ancestor a, the level in the tree l of a (the root is at level 1; the tree is 5 level deep), the distance d_1 between a and t_1 and the distance d_2 between a and t_2. We chose a function so that the higher d_1 and d_2 the lower the similarity should be, and the closer to the root of the tree

a is, the lower the similarity should be. We chose to use the following function: $sim(t_1, t_2) = (.15 * (l))^{d_1+d_2-1}$.

In the ACM/IEEE 2013 curriculum guidelines, most entries picked by a user would be at level 4. Two topics that are siblings at level 4 (in the same Knowledge Area and the same Knowledge Unit) would have a similarity of 0.45, while two cousins at level 4 (in the same Knowledge Area and different Knowledge Units) would have a similarity of 0.02.

To compute the similarity of two materials m_1 and m_2, we build a bipartite graph of all the classification entries in m_2 and connect them to all the classification entries in m_2. Each edge (t_1, t_2) is weighted by $sim(t_1, t_2)$. We compute a maximum bipartite matching which identify the best way to match each entry of m_1 to an entry of m_2. Finally, the value of the matching is normalized by the number of classification entries of m_1 and m_2.

4.3.2. Visualization of search results

A common problem of search engines is that it is hard for a user to tell at a glance how good a result is and how the results relate to each other. Fortunately we can leverage similarity calculation to make useful visualizations. For a set of materials (we consider the query as a material for search operations), we create a graph where the materials are vertices of the graph. Each edge between two materials is weighted by the similarity between the two materials. In practice, to avoid a cluttered display we only keep the edges of the graph with highest weight. The number of edges we display is 3 times the number of vertices.

The similarities are passed to a Multidimensional Scaling (MDS) algorithm [37] to obtain for each vertex a location in a 2D space. The opacity of the graph's edges are made proportional to the edge weight normalized by the edge of highest weight to visually hint at the similarity between the materials.

5. Using CS Materials to Improve PDC Adoption

The types of usage one will have of CS Materials will depend on who the user is and what type of task they are trying to accomplish. We will focus on two

hypothetical users: an instructor of an early CS class who wants to integrate some PDC content in his/her class, and a PDC educational expert looking to help the broader CS community adopt PDC content in their courses.

We have used CS Materials to derive an understanding about various courses and materials. To date, we have added two different sections of a required data structure class taught at UNC Charlotte, a capstone software development class taught at UNC Charlotte, an elective parallel computing class taught at UNC Charlotte, an elective class on object-oriented programming, about 65 Nifty assignments [12], and all Peachy assignments [9] prior to 2019. Two courses have been added (one Algorithms course, and one CS 1 course) by users from other institutions. Three additional courses, ranging from CS1 to Data Structures, are also in the process of being added to the system by instructors from other universities.

What we present here are our initial experiences of how we have used CS Materials, what is involved in using it, and how it can be used to get more adoption of PDC content.

5.1. Understanding the Guidelines by Entering a Parallel Computing class

We classified ITCS 3145:Parallel and Distributed Computing, taught at UNC Charlotte [13]. Materials in this class are composed of 12 slide decks and 9 assignments. Inputting (including classifying) all the materials took the instructor of that class (one of the authors, Dr. Saule) about a day of work, with each item taking between 15-25 minutes to input and classify. The classification of these three classes of material can be seen in Figure 4.

Keying the meta data is straightforward and fast, but classification against curriculum guidelines is more involved, because of the size of the ontologies (the CS13 classification contains about 3000 entries). One could quickly make some selections, but most likely result in missing relevant entries. For instance, in the ACM CS13 guidelines, parallelism related topics appear in three different places: *System Fundamental*, *Computational Science::Processing*, and in *Parallel and Distributed Computing*. Going quickly through the classification would most

likely result in a poor classification of the material.

In the PDC12 guidelines, some odd placements also exist; for instance, Amdhal's law and related topics fall under *Programming::Performance Issue::Data. Algorithm::Model based notions::Parallel and Distributed Models and Complexity::Notions from scheduling* misses Critical Path. The Map-Reduce programming model seems mostly missing. (There are entries for BSP; which is oddly bundled with Cilk and Cloud Computing but these are not quite the same). Overall, the PDC12 guidelines were a first attempt at classifying PDC topics, and the 2020 edition of PDC is expected to correct these oddities.

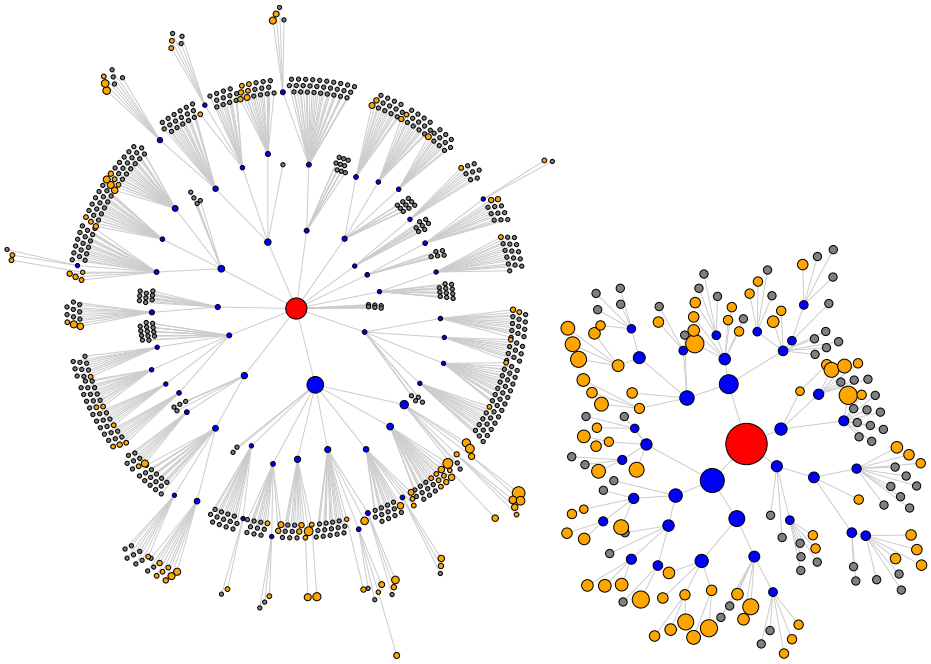
In both classifications, topics related to middleware (design and implementation) seem to be mostly missing. Runtime systems appear under Programming Languages in CS13, but refer to different things. Also on many topics, both classifications seem to stay at a high level. While this is appropriate for curriculum guidelines, it is not as precise for classification of material as one would hope for. For instance, ACM CS13 has an entry for Task-Based Decompositions, but recursive Cilk-style decomposition are different from OpenMP depends-style decomposition.

The complexity of the classification makes it easy for one to misclassify some materials. Using the harmonization view (described in section 4.1.3) enables a user to easily check whether the materials were classified consistently. For instance, all OpenMP materials should probably be classified to match "compiler directives/pragma". However in the first pass of classification, not all OpenMP materials were classified with that curriculum entry, that was then corrected using the harmonization view.

Also it seems that it would be useful not only to say that a material matches a topic, but at which level the topic is matched. Taking an example, an early assignment in ITCS 3145 was to implement a numerical integrator using the rectangle method. Naturally this assignment checks *Computational Science::Numerical Analysis::Numerical differentiation and integration.* But the assignment only covers integration and only requires the students to implement a single method from a provided formula. A numerical methods course would

have a more comprehensive lecture on numerical integration and would check the box in the same way. Since both ACM CS13 and PDC12 guidelines have incorporated Bloom levels [28], it would make sense to classify materials with Bloom levels as well.

Overall, the experience of inputting and classifying a full course, though somewhat labor intensive, allows an instructor to reflect on the content and structure of the course and assess it against national guidelines. This frequently leads to reconsider the course design and content; this was was our experience with a data structures courses that was input by one of the coauthors (Dr. Subramanian), and detailed in an upcoming publication [14].



(a) IEEE/ACM CS 2013 (b) CDER PDC 2012

Figure 4: ITCS 3145 classification against two curriculum guidelines.

5.2. Searching for Materials to Integrate in a Data Structures class

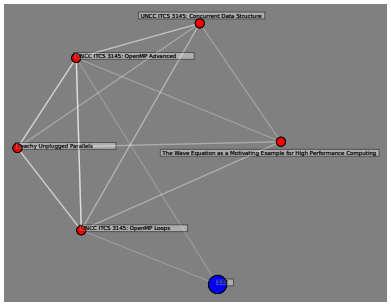
A strategy to bring Parallel and Distributed Computing content into existing curriculum is to add a core course in a degree program. This solution is usually

not favored in the US because it seems difficult to implement. We believe it is more scalable to splice the PDC content in early CS courses that are already taught such as CS1, CS2, Data Structures, Computer Architecture, etc. Each course can then present what they would normally present, but expanded with PDC content that is appropriate for the topics and level of the class. We present here such an analysis for a Data Structures course. We believe Data Structures is a good candidate because it appears in all programs under one form or another (with CS1 and CS2), it is composed of topics that are not naturally presented as parallel or distributed (unlike Computer Architecture or Operating System), there is little variance in how the class is taught (CS1 on the other hand has many variants including object-first, imperative-first, and functional-first approaches), and we have two fully classified Data Structure classes in CS Materials.

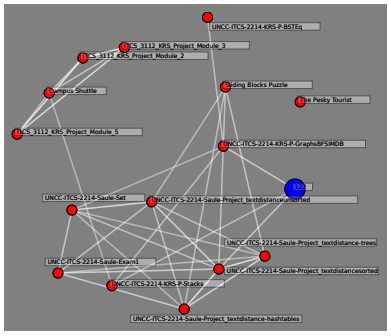
We picked four materials from data structure classes and used the search feature of CS Materials to identify similar materials within all the material available in CS Materials, and within PDC materials in particular. The goal of the search is to find replacement materials or materials that could be used to expand on the topic. The four materials we selected are a project on linked list, a lecture on Big-Oh notation, a lecture on Shortest Path, and a homework on binary trees. We selected these materials because they span the main components of what we perceive to be a canonical data structures course: complexity, linear structure, hierarchical structures, and graphs.

The search results are displayed in Figure 5; we performed the search against PDC materials to only include the top-5 results, while the search against all materials use the top-15 results. Note that opacity of edges in the different pictures relates to the similarity between the materials connected by the edge; however different pictures see the similarity normalized differently and as such the opacity of edges in different figures are not directly comparable.

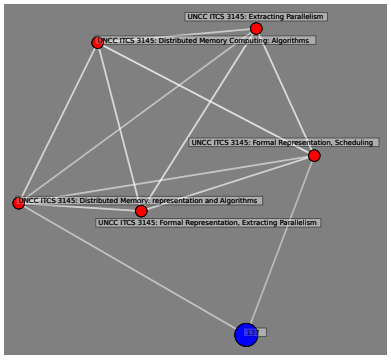
Finding materials related to a Linked List project and covering PDC content yields two OpenMP lectures, which do not contain linked list concepts, two Peachy assignments which also do not contain linked list, and one lecture on



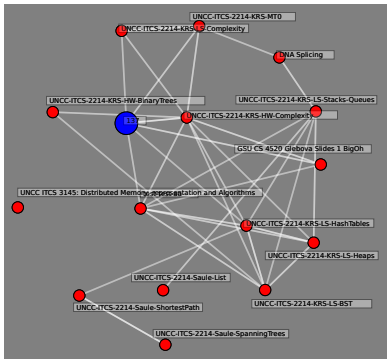
(a) Linked list project against PDC materials



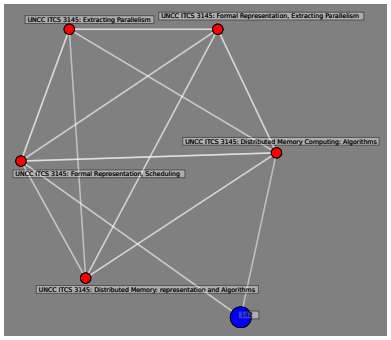
(b) Linked list project against all materials



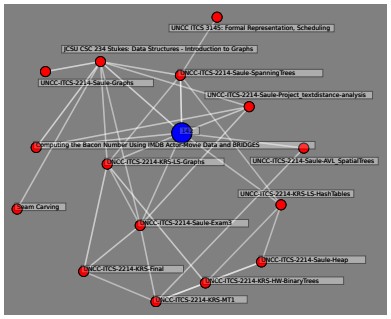
(c) Big-Oh lecture against PDC materials



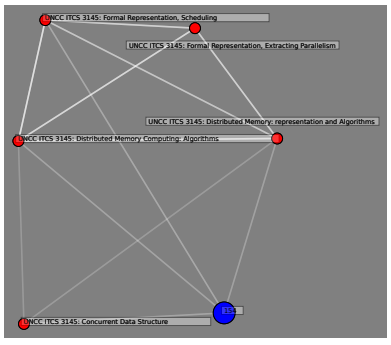
(d) Big-Oh lecture against all materials



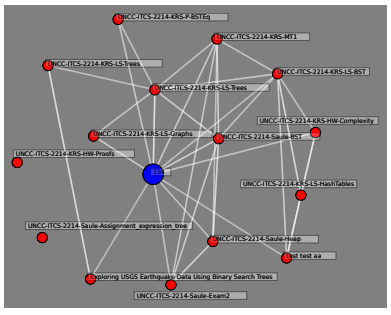
(e) Shortest path lecture against PDC materials



(f) Shortest path lecture against all materials



(g) Homework on binary trees against PDC materials



(h) Homework on binary trees lecture against all materials

Figure 5: Searching for replacement materials for a few topics in CS Materials.

concurrent data structure which uses linked list as an example of lock-free data structures. The PDC materials related to the original project are underwhelming. However, one can see that there are good recommendation in the general pool of materials: the recommended text distance projects are implementations of dictionary interface using various data structures including a linked list, as well as lectures and an exam that contain linked list related materials. One of the three Nifty assignments recommended using a Linked list. The cluster of ITCS 3112 projects however do not have linked list content.

The Big-Oh lecture yielded PDC results from a parallel computing class, that were from two lectures, three assignments. All these use Big-Oh notations or fundamental algorithmic tools to discuss resource consumption of parallel application. While none of the material would be directly usable to discuss Big-Oh in a data structure level course, the content could be used to give example usage of Big-Oh in a parallel computing context, to help students understand that not only runtime is expressed as Big-Oh. Using the same Big-Oh lecture as a query against all materials yields one of the PDC materials. Most importantly, CS Materials suggests alternative lectures, homework, and exam around Big-Oh. The system also suggest a trove of data structure lectures which use Big-Oh notations without Big-Oh being the primary topic.

The shortest path lecture yields no directly useful PDC materials. We would expect to be recommended a lecture on parallel shortest path algorithms, or shortest path in a distributed context such as in networking protocols. The materials that get recommended are all parallel algorithm analysis which use graph abstractions, but none that actually do any shortest path computations. The general materials that get recommended are all graph and tree materials. Only one of them does a weighted shortest path, though a few of the graph related materials (including assignments) discuss BFS traversal which is fundamentally an unweighted shortest path.

The final query looks for materials similar to a homework on binary trees. The recommended PDC materials are once again the theoretical parts of a parallel computing class which are not particularly relevant, however it also

recommends the lecture on concurrent data structures which uses binary search trees as an example. The general materials that get recommended are much more appropriate. Lectures, assignments, and exams on different usage of binary trees get returned including heaps, binary search trees, and spanning trees. Some other non-tree data structure also get returned.

Overall, the study of the search feature of CS Materials show that the system has the ability to find related materials. Though of course, CS Materials can only recommend the materials that are currently in the system.

5.3. Are Nifty assignments like Peachy assignments?

Peachy assignments were created as a way to showcase high quality assignments which are easily adoptable to teach topics in parallel and distributed computing. The assignments are meant to be patterned after Nifty assignments which focus on early CS courses such as CS1, CS2, and data structure courses.

We study here whether Peachy assignments can be used instead of Nifty assignments as one of the means to introduce PDC content in early CS courses. We use the radial view to showcase the difference between these two sets of materials. The generated visualization is shown in Figure 6. In this difference view, the inner nodes are the ACM classification entries of the two sets, the leaf nodes which appear white are common between Peachy and Nifty assignments, the purple ones are only contained in Peachy assignments, while the orange nodes are Nifty assignments.

Very few leaf nodes are white, which indicates that the two sets mostly map to different parts of the curriculum guidelines. The area of maximum overlap is in Software Development Fundamental, more precisely the topics relating to array processing. Indeed, it is a common topic of CS1, and many Peachy assignments are essentially and pleasingly parallel for computations.

Interestingly, the two sets map not only to different topics of the curriculum guidelines, but also to different sections of it. Peachy assignments are almost the only mapping to Parallel and Distributed Computing, Operating System, System Fundamental, and Architecture and Organization. These sections of the

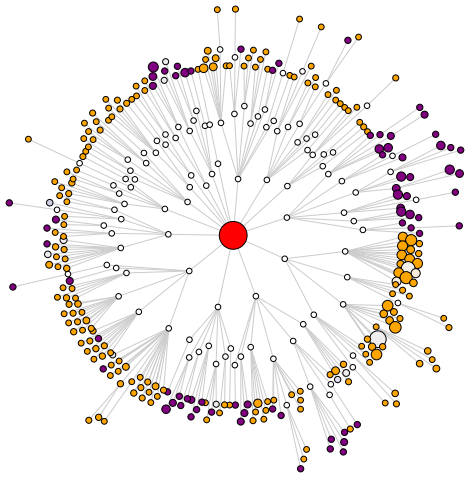


Figure 6: Difference of curriculum mapping between Nifty assignments and Peachy assignments against the ACM/IEEE CS 2013 guideline. The inner nodes are part of the ACM curriculum entries, the leaf nodes are either purplish (Nifty only), orange (Peachy only) or white (common to both).

ACM/IEEE CS 2013 are essentially not touched by Nifty assignments.

On the other hand, Nifty assignments map more broadly to the curriculum guidelines. Nifty assignments reach a larger coverage of topics in Computational Science, Algorithms and Complexity, Programming Languages, and Human Computer Interaction.

This analysis shows that while a meaningful effort, *Peachy assignments in their current form, will probably not easily be adopted in CS1 and CS2 courses.*

5.4. Identifying Missing Materials to Integrate PDC content in Data Structures Classes

One of the primary hurdles of attempting to integrate PDC content into early CS curriculum is that we may not know precisely what these early courses cover and what they teach. There are lots of different ways to teach CS1, not only in the topics covered, but also in pedagogical styles, activities, collaboration, etc. CS Materials provides us with an opportunity to uncover these differences and build models of these courses to help PDC expert target their materials to these

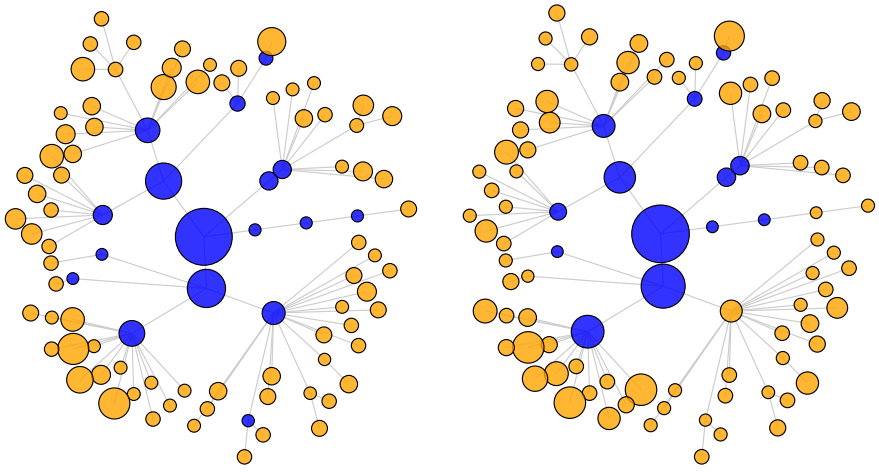
commonly covered topics. We demonstrate this with a data structures course model.

5.4.1. How to define Data Structures?

We built a preliminary model of what a typical data structure course might contain. We compute a model of the topic of a single course by taking the union of all the classification items of the materials of that course. And we compute a model of data structure as an abstract class by taking the intersection of the classification items of all the data structures courses available in CS Materials.

Obviously, this model for data structures is very simple but it is appropriate since there are only two data structures courses in CS Materials at the moment. With more data structure courses, a more refined model can be designed. Yet this simple model provides a reasonable list of topics which are listed in Table 1.

We generated coverage views of the modeled data structure topics and learning outcome for both the data structure courses used to build the model, as shown in Figure 7. One can see that the two courses cover the data structure topics in similar proportion, confirming the previous analysis that the course appeared to agree on the coverage of main data structure topics.



(a) Saule's ITCS 2214 (b) Subramanian's ITCS 2214

Figure 7: Coverage of data structure topics by two data structures courses

Trees
Properties
Traversal strategies
Undirected graphs
Directed graphs
Weighted graphs
Spanning trees/forests
Use for generic libraries such as collections
Asymptotic analysis of upper and expected complexity bounds
Big O notation: formal definition
Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential
Empirical measurements of performance
Time and space trade-offs in algorithms
Big O notation: use
Little o, big omega and big theta notation
Analysis of iterative and recursive algorithms
Balanced trees (e.g., AVL trees, red-black trees, splay trees, treaps)
Greedy algorithms
Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
Minimum spanning tree (Prim's and Kruskal's algorithms)
Heaps
Sequential and binary search algorithms
Worst case quadratic sorting algorithms (selection, insertion)
Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort)
Hash tables, including strategies for avoiding and resolving collisions
Binary search trees
Common operations on binary search trees such as select min, max, insert, delete, iterate over tree
Graphs and graph algorithms (Tier 1)
Depth- and breadth-first traversals
Representations of graphs (e.g., adjacency list, adjacency matrix)
Iterative and recursive traversal of data structures
Expressions and assignments
Simple I/O including file I/O
Conditional and iterative control structures
The concept of recursion
Arrays
Records/structs (heterogeneous aggregates)
Strings and string processing
Abstract data types and their implementation
Stacks
Queues
Priority queues
Maps
Linked lists
Strategies for choosing the appropriate data structure

Table 1: Model of Topics in a typical Data Structure course. For clarity, we removed the Learning Outcome and removed the Knowledge Unit and Knowledge Area prefixes.

5.4.2. How good is coverage of Data Structures courses by PDC materials in the system

Provided a model of a class, we can study the coverage of PDC materials that aligns with that class. Here we used the model of a typical data structure course to study the coverage of PDC courses in a data structure course. This analysis is presented in Figure 8. What the analysis show is that while some topics are mapped to by some PDC materials, most are not.

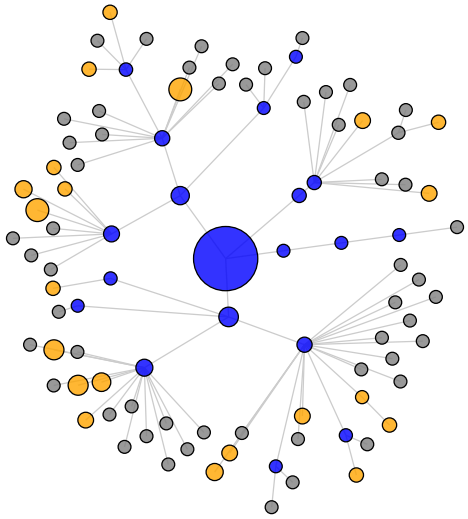


Figure 8: Coverage of modeled data structure classification item by all PDC materials in CS Materials

The most meaningfully mapped topics related to array processing and iterative control structure. They are covered by a few of the Peachy assignments, mapping to parallel array processing. Classification items related to Queues, Maps, Hash Tables, Binary Search Trees are matched to a lecture on concurrent data structures from ITCS 3145. Topics on Hashing and Hash tables also match a lecture on Map Reduce.

Concepts on recursion are mapped to a lecture on OpenMP's tasking construct which highlight that one could use OpenMP for implementing Cilk-like parallelism. Sorting concepts are mapped to a couple assignment and an unplugged activity.

The most unexpected matching came from theoretical concepts of a data structure such as Big-Oh notations, which appear in discussion of parallel complexity, communication complexity. For the same reason lectures on task scheduling map to topics of graphs (for PTGs) and of greedy algorithms (for List Scheduling).

It is also interesting to note that many topics in data structures do not appear to be covered by materials mapped to PDC topics in CS Materials such as topics around linked list, structures, priority queues, stacks, strings, tree orderings, spanning trees, modeling, library use, complexity classes, analysis of data structures. The authors believe that all these topics could be covered by PDC materials, but happen not to be. It is also note worthy that topics that are matched, are typically matched by only a single material, leaving little choice for potential instructors looking to adopt PDC content.

6. Evaluation: Early User Feedback

We conducted a CS Materials workshop in summer of 2020, where we presented an overview of CS Materials, its features and benefits to CS educators. The workshop had 13 attendees. Four of them followed up with a more hands-on workshop where they input and classified a few of their course materials over a 2 hour session. The goal was to complete the entire course on their own time after the workshop. Two of the instructors completed classifying their courses and, in addition, filled out a survey of their experience in using CS Materials. The survey had 8 questions, relating to UI (usability, navigation), usefulness of the visualizations (radial view and the harmonization view) to gain insights, time taken to complete the classification, and a final overall impression of CS Materials. The courses that were classified by the two users were on Programming Concepts and Methodology I (equivalent to CS1) and Design and Analysis of Algorithms.

Overall, feedback from both instructors was positive. They found the system somewhat easy to navigate and classify their materials, and spent 6-15 hours to

add and classify their materials on the site. The CS1 course had 13 sets of lecture slides, 10 homeworks and assignents, and 6 exams. The Algorithm Analysis course had 10 sets of lecture slides, 8 homeworks, assignments and reflections, and 5 exams. Amount of time spent on the assignments depends on the number and complexity of the materials, however, the classification process usually goes faster as the user becomes more familiar with the curriculum standards. One user did not gain any insights during the classification process ("can't say that during classification process I was thinking holistically yet. I was really just trying to find where in the classification each material belongs to"). The other noted gained insight about the course while classifying ("[M]y image processing assignment [...] covers a pretty wide breath of topics, and so I think it could almost could be used as a capstone assignment for the assignment, though I usually issue it about halfway through the semester.")

Both users benefited from the radial view ("weighting of my class in the radial view matches what my mental model of the weighting should be", "it is easier for me to see how topics coverage is spread out out of ACM classification") and one user "made changes by adding topics I missed during classification". Feelings on the the harmonization view were mixed: one user found it beneficial ("really interesting from the perspective of being able to view all the content in my course from several different angles", while the second user found it somewhat unwieldy ("I found matrix (harmonization) view less useful for me to see overall coverage and mapping"). More work is needed for the harmonization view as it compresses a significant amount of information in a single visualization.

The overall impression of CS Materials from the two users was quite positive. Users touched on the value of CS Materials with comments that focused on definition of a specific course, "Figuring out what people actually mean by 'teaching CS1' and what, exactly, they are teaching, with what topics, in what order, with assignments, is amazingly valuable", on comparing courses, "it would be very interesting to compare courses of the same kind (CS1, Algorithms, etc) with others" and deriving benefits from such analysis, "This way I'd get an insight

whether we spend too much time on particular topic and no one else does? Or vice versa". One suggestion was to also capture different pedagogies as part of the classification, "What I found missing is that there is no way to say what kind my class is in terms of active learning strategies or teamwork, hybrid/flipped modalities or otherwise". We will investigate these issues, as these pedagogies have become quite popular in recent years.

7. Discussion

The overall motivation for the CS Materials was driven by the larger goal of producing a highly dynamic system that would house a variety of course designs and content that could be efficiently searched, analyzed and shared among instructors, program administrators, as well as for audit and assessment. On the other hand, we did not want CS Materials to suffer the same fate of many of the existing repositories due to lack of maintenance, support, etc. Thus, our approach to continually align with nationally accepted standards serves as a bulwark to keeping the content current, relevant and responsive to new technologies. In this work, we adapt these goals towards instructors to adopt PDC content, by building a system that makes it easier and efficient to find new or equivalent materials; at the same time the system helps identify gaps in PDC content, that can be valuable for the PDC community for developing new materials.

CS Materials is in its early stages of development. The system is functional and ready for use by instructors as well as the PDC community; the main hurdle is to get a critical mass of content and users. As stated earlier, we have a few courses in the system and materials relevant to both early CS courses as well as some PDC content. However, even with this limited amount of content, we have demonstrated the potential of the system to search and identify materials by using ontology based searches, compute similarity between sets of materials, and generate course models for use in comparative analyses. Obviously to derive better insight, one would need more data. While it is not clear at this point how

many courses would need to be classified to answer a question like "how many formats of a data structures course exist?", however, a system like CS Materials appears to be the only way to be able to provide an answer to such a question.

As CS Materials is based on using classification tags as the basis for various analytic features that we have presented in this work, our future plans include keeping the system updated as the standards change; typically these updates occur every 5 years (or more), and would require a one time update of the system to conform the new standards and conversion of the materials in the system to the new standard, which we believe can be automated.

As mentioned earlier, instructors themselves have a number of advantages in using the system for their own course development, keeping their content current and searching for suitable materials for incorporating PDC content. Similarly, degree auditors can benefit from the current system, as they can input an entire sequence of courses into the system and perform audits, check proper sequencing of topics/learning outcomes, and ensure incoming students have the right prerequisites. While the illustrated results in the current system are somewhat underwhelming, this is primarily due to the limited amount of content in the system. We believe that once we get a sufficient amount of materials populated in the system, more ambitious queries and searches would result in more relevant content.

The experiments on looking for coverage of PDC content for use in early CS courses yielded mixed results. This could be due in part to the few materials that are currently in the system. But the authors manually looked through the PDC material they are aware of (and listed in Section 2) and do not believe the results would be significantly different, even if they had been added to the system. Materials in CDER courseware, or CS in Parallel that are identified as appropriate for "data structure and algorithms" would not have appeared for two reasons. First, some of the materials are appropriate for the technical skills of a student taking data structure but the materials do not integrate well within the topics of data structures. Second, many PDC materials for a "data structure and algorithm" course are about divide and conquer algorithms

and recursive Cilk-like parallelism. These topics would not be taught in a "data structure" course; rather that assumes a separate "algorithms" course will follow in the curriculum. Similarly, we did not find assignments on producer-consumer systems listed as appropriate for "data structures", despite they fundamentally rely on concurrent queues. Producer-consumer assignments were only classified as appropriate for an "operating systems" class probably because that is the class where they are traditionally taught. This bolsters our claim that one should classify against topics and let classes be an emergent concept.

Overall, our study highlights the need for the PDC community to develop new content. Ideally one would work together with instructors of targeted courses for maximum impact. The main novelty in using CS Materials for this purpose is that the type of content that needs to be developed is much clearer in terms of the topics and learning outcomes; in other words, the needs are identified at a more granular level. Also, CS Materials enables the effort to be easily discovered by instructors without the need to pair with a PDC expert.

Our own experience in entering courses into the system has also significantly enlightened us in terms of the content, design and learning outcomes. While this is not necessarily the primary goal of this work, it provides an opportunity for course revisions and improvement that cannot be understated. At the same time, instructors become more familiar with curriculum guidelines and begin to think of their course designs along with more current content.

8. Conclusion

The efforts to include PDC content in early CS courses, such as incorporating them in national curriculum guidelines have so far been limited. A commonly cited reason is that instructors have difficulties finding relevant material to teach and learn these topics. In this article, we argue that classifying learning materials against standard ontologies such as PDC12 and CS13 and curating them in a single system provides many advantages for multiple actors. We have demonstrated this by building CS Materials, a system for classifying, searching and

performing analyses of materials in the system. We have demonstrated a number of features in the system, including an ontology based search, determining the coverage of the guidelines of sets of materials, measuring alignment between sets of materials, including alignment, and ability to build models of courses coming from different sources. While the amount of content in the system is limited at this time, initial tests of the system demonstrate its potential for the long-term. Future work will focus on publicizing the system to potential users across the education community to enable instructors to adopt more PDC materials.

Acknowledgments

The authors would also like to thanks Nick Parlante for his work on curating the Nifty assignments and David Bunde for his work in curating the Peachy Parallel assignments.

This work is supported by grants from the National Science Foundation (CCF-1652442, DUE-1245841, and DUE-1726809) and by a summer undergraduate research fellowship from the Charlotte Research Scholars Program.

References

[1] NSF/IEEE-TCPP Curriculum Working Group, NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing : Core topics for undergraduates, Tech. rep., CDER, available at `http://www.cs.gsu.edu/~tcpp/curriculum/sites/default/files/NSF-TCPP-curriculum-version1.pdf` (2012).

[2] Joint Taskforce on ACM Curricula, Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, ACM/IEEE Computer Society, 2013 (2013).
URL `https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf`

[3] cc2005, Computing curricula 2005: The overview report, https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2005-march06final.pdf.

[4] cc2020 pdf, Computing curricula 2020: Paradigms for future computing curricula, https://cc2020.nsparc.msstate.edu/wp-content/uploads/2020/11/15September2020-CC2020-Report-v43.pdf.

[5] J. Impagliazzo, A. N. Pears, The cc2020 project — computing curricula guidelines for the 2020s, in: 2018 IEEE Global Engineering Education Conference (EDUCON), 2018, pp. 2021–2024 (2018). doi:10.1109/EDUCON.2018.8363484.

[6] A. Clear, A. Parrish, M. Zhang, G. C. van der Veer, Cc2020: A vision on computing curricula, in: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, SIGCSE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 647–648 (2017). doi:10.1145/3017680.3017690.
URL https://doi.org/10.1145/3017680.3017690

[7] S. Prasad, A. Gupta, A. Rosenberg, A. Sussman, C. Weems (Eds.), Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses, Morgan Kaufmann,, 2015 (2015).

[8] S. Prasad, A. Gupta, A. Rosenberg, A. Sussman, C. Weems (Eds.), Topics in Parallel and Distributed Computing: Enhancing the Undergraduate Curriculum: Performance, Concurrency, and Programming on Modern Platforms, Springer International Publishing, 2018 (2018).

[9] Peachy parallel assignments, https://grid.cs.gsu.edu/ tcpp/curriculum/?q=peachy.

[10] R. Brown, L. Shoop, J. Adams, CS in parallel, https://csinparallel.org/.

[11] S. Matthews, Pdc unplugged: A free repository of unplugged parallel & distributed computing activities, https://tcpp.cs.gsu.edu/curriculum/sites/default/files/EduPar20.pdf.

[12] N. Parlante, Nifty assignments.
URL http://nifty.stanford.edu/

[13] E. Saule, Experiences on teaching parallel and distributed computing for undergraduates, in: Proc of IPDPSW 2018, 2018 (May 2018).

[14] A. Goncharow, M. Mcquaigue, E. Saule, K. Subramanian, J. Payton, Mapping materials to curriculum standards for design, alignment, audit, and search, in: Proc. of SIGCSE 2021, 2021, (to appear) (2021).

[15] AAAI, Model AI assignments.
URL http://modelai.gettysburg.edu/

[16] Groovy Graphics Assignments, https://blog.siggraph.org/tag/groovy-graphics-assignments/ (Accessed July 2019).

[17] E. Ayguade, LlucAlvarez, F. Banchelli, M. Burtscher, A. Gonzalez-Escribano, J. Gutierrez, D. A. Joiner, D. Kaeluu, F. Previlon, E. Rodriguez-Gutiez, D. P. Bunde, Peachy parallel assignments (EduHPC 2018), in: Proc. of EduHPC, 2018 (2018).

[18] A. Monge, B. A. Quinn, C. L. Fadjo, EngageCSEdu: CS1 and CS2 materials for engaging and retaining undergraduate CS students, in: Proc. of ACM SIGCSE, 2015, pp. 271–271 (2015).

[19] NCWIT. [link].
URL https://www.engage-csedu.org/

[20] G. Sprint, A. O'Fallon, Engaging programming assignments to recruit and retain CS0 students: (abstract only), in: Proc. of ACM SIGCSE, 2018, pp. 1093–1093 (2018).

[21] A. C. Bart, CORGIS Datasets Project: The Collection of Really Great, Interesting, Situated Datasets, https://think.cs.vt.edu/corgis/.

[22] A. Bart, E. Tilevich, S. Hall, T. Allevato, C. Shaffer, Transforming introductory computer science projects via real-time web data, in: Proc. of ACM SIGCSE, 2014, pp. 289–294 (2014).

[23] A. C. Bart, R. Whitcomb, D. Kafura, C. A. Shaffer, E. Tilevich, Computing with CORGIS: Diverse, real-world datasets for introductory computing, ACM Inroads 8 (2) (2017) 66–72 (Mar. 2017).

[24] D. Burlinson, M. Mehedint, C. Grafer, K. Subramanian, J. Payton, P. Goolkasian, M. Youngblood, R. Kosara, BRIDGES: A system to enable creation of engaging data structures assignments with real-world data and visualizations, in: Proc. of ACM SIGCSE 2016, 2016, pp. 18–23 (2016).

[25] K. Subramanian, BRIDGES (Bridging Real-world Infrastructure Designed to Goal-align, Engage, and Stimulate),.
URL `http://bridgesuncc.github.io/`

[26] A. Decker, M. M. McGill, L. A. DeLyser, B. Quinn, M. Berry, K. Haynie, T. McKlin, Repositories you shouldn't be living without, in: Proc. of ACM SIGCSE, 2018, pp. 920–921 (2018).

[27] M. Leake, C. M. Lewis, Recommendations for designing CS resource sharing sites for all teachers, in: Proc. of ACM SIGCSE, SIGCSE '17, 2017, pp. 357–362 (2017).

[28] B. Bloom, M. Engelhart, E. Furst, W. Hill, D. Krathwohl, Taxonomy of educational objectives: The classification of educational goals, David McKay Company, 1956 (1956).

[29] National Security Agency, Centers of academic excellence in cyber defense (CAE-CD) – 2019 knowledge units, Tech. rep., NSA (2018).

[30] College Board, Computer Science A: Course Description, College Board AP, Fall 2014 (Fall 2014).
URL `https://apcentral.collegeboard.org/pdf/ap-computer-science-a-course-description.pdf`

[31] College Board, AP Computer Science Principles, Including the Curriculum Framework, College Board, Fall 2017 (Fall 2017).

[32] M. Tungare, X. Yu, W. Cameron, G. Teng, M. A. Pérez-Quiñones, L. Cassel, W. Fan, E. A. Fox, Towards a syllabus repository for computer science courses, in: Proc. of ACM SIGCSE, SIGCSE '07, 2007, pp. 55–59 (2007).

[33] Joint Taskforce on Computing Curricula, Computing Curricula 2001 Computer Science, ACM/IEEE Computer Society, 2001 (2001).
URL http://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf

[34] D3: Data Driven Documents. [link].
URL https://d3js.org/

[35] K. Eren, M. Deveci, O. Küçüktunç, Ü. Çatalyürek, A comparative analysis of biclustering algorithms for gene expression data., Briefings in Bioinformatics 14 (3) (2013) 279–292 (May 2013). doi:10.1093/bib/bbs032.

[36] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, no. ISBN 0-321-32136-7, 2005 (2005).

[37] Modern Multidimensional Scaling: Theory and Applications, 2nd Edition, Springer-Verlag New York, 2005 (2005). doi:10.1007/0-387-28981-X.