UBCourses Visualization

Siyuan He
hesiyuan@cs.ubc.ca
Jiahong Chen
jhchen@mech.ubc.ca

November 2017

1 Data and Task Abstractions

- **Domain**: Course Information from The University of British Columbia.
- **Task**: Specific tasks depend on the goal of people using it. There are many different tasks. These include but not limited to (a) Students who want to know if they satisfy all prerequisites for a course; (b) Curriculum planners who want to analyze dependencies of courses across departments; (c) Students who wish to plan their courses towards their degree.
- **DataSet**: The dataset we extracted consists the credits, prerequisites, co-requisites and equivalent courses of all UBC courses. For now, we narrow our infovis solution towards to mainly prerequisites. The raw dataset in pdf version is available through UBC calendar website and will be updated every year. The following shows the three types of prerequisites in the UBC course catelogue.
  1. Mandatory: All these courses must be satisfied before taking the desired one.
  2. One of: At least one of these courses should be taken before enrolling.
  3. Either or: Either these courses or other course(s) should be taken.

2 Proposed Infovis Solution

Course information provided either by UBC calendar or UBC course registration system is limited to the amount of information it conveys at a time. Particularly, some problems arise if one is trying to click on a series links of prerequisites. Especially if one is interested in the overall structure of courses and we see that it's hard to get this information solely through a table of courses or HTML links. Thus, we present two views for visualizing courses. For people who are curious in the topology or dependencies of courses, we present a constrained directed graph where each node represents a course, and each directed edge denotes a prerequisite relation. In addition, we shall also use hue-coded containment to encode various options in satisfying prerequisites. For people who are interested checking if they have obtained all prerequisites for a course or a set of courses, we present a novel approach where horizontal and vertical positions encode meaning of courses. The above two views are linked through the use of multiform overview/detail idioms.

2.1 Topology View

2.1.1 Encoding for prerequisites

Encoding for *mandatory* courses is easy. If course A must be satisfied before taking course B, then a directed edge from A to B is drawn. If more than one *mandatory* courses exist for a course, then each of these courses is drawn an directed edge to that course. However, if a course's prerequisites pose

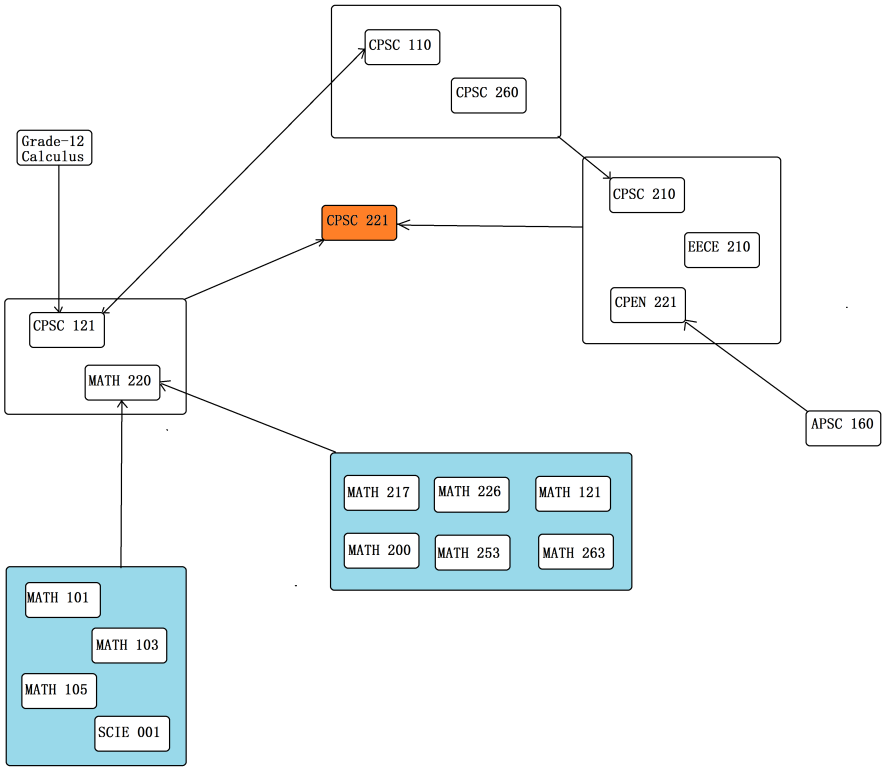

Figure 1: Topology View of Courses Using Containers and Directed Edges

an option that one of the several courses must be taken, we first draw a container which consists of these courses and then have a directed edge from the containment to that course. An Example of this encoding is shown in CPSC 221. It can also happen that of the rules entered by course registration staff multiple *either or* prerequisites exists for a single course. This case is a natural generalization of the *one of* prerequisites, and we shall use one container with a hue for each *either or*. Since the number of *either or* of a given course is usually around one or two, categorical hues can be used efficiently.

The topology view can also help the department to schedule courses better. By visualizing whole department's courses and their interaction with courses in other departments, faculties and student services could have a better understanding of the dependencies of courses being offered. Then, outdated courses can be replaced by more suitable ones. Besides, coordination with other departments can be improved since there might be some duplicated courses offered by the different department. Having the whole network of provided courses can help the decision maker to provide better study plans for students.

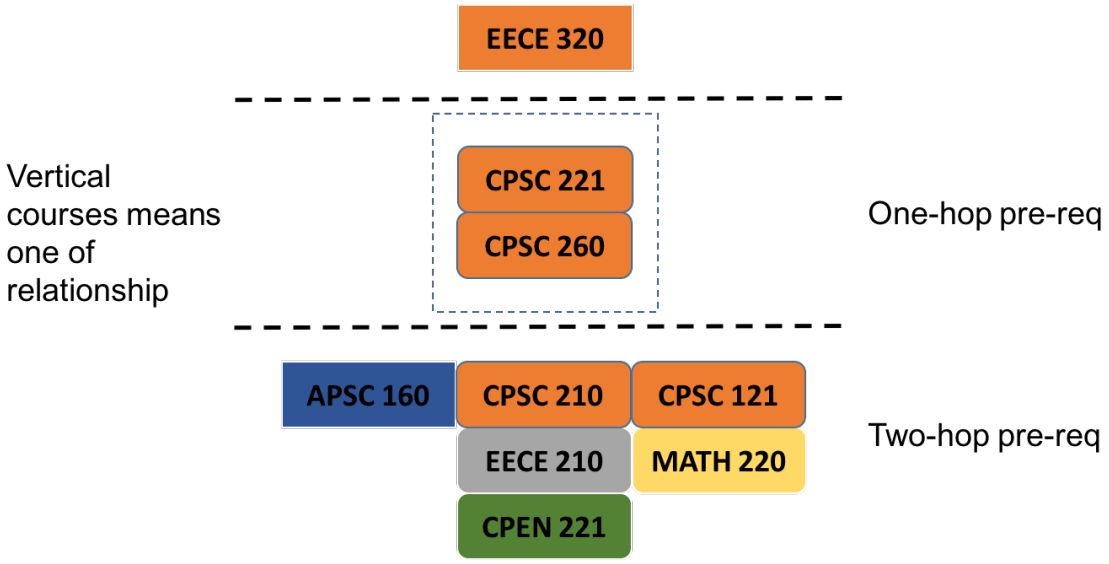

Figure 2: Example of single course prerequisite visualization

2.1.2 Encoding for co-requisites, equivalencies, and soft-wired prerequisites

The encoding for prerequisites allows us easily to extend to co-requisites. If two courses are co-requisites, then two directed edges are drawn in opposite directions. For courses that are equivalent, we shall extensively use containers. As for soft-wired prerequisites, for example, a third-year standing, it is out of the scope of this project. However, we can trivially satisfy them by drawing a dummy node with their corresponding labels.

2.2 Student View

2.2.1 Encoding for prerequisites

In this project, we consider two student view: a specific course, and courses taken for satisfying the degree. The basic view is visualizing a prerequisites tree for a specific course, especially for junior undergraduate students who want to take a specific higher level course in the future and trying to register its prerequisite courses now. Usually, multiple prerequisites exists for a 300-level or 400-level course and these prerequisites, in turn, have their prerequisites. Moreover, these prerequisites might have other relationships such as overlapping courses, mandatory courses, and optional courses. Thus, it makes the situation much more complex when traveling this prerequisite tree from top to bottom. To help students focus on what are the necessary prerequisites for a specific course, a good idea is using our interactive course visualization view.

Figure 2 is an example of how we might visualize prerequisites of a specific course, assuming we want to know the prerequisites of the EECE 320. Top level states the main course that we want to take, followed by its immediate or one-hop prerequisites. In the one-hop prerequisite level, all prerequisites required by top level courses should be presented. Courses presented vertically at this level means only one of these courses should be satisfied. Next, two-hop prerequisite works similarly. Horizontally presented courses mean all of them should be satisfied if all courses at the previous level are selected. Since not all courses are necessary, users can make their own choice and all irrelevant courses will disappear. As shown in Figure 3, if the user plans to select CPSC 221 as the prerequisite of EECE 320, unnecessary course CPSC 260 and its dependent prerequisite will become transparent. Then, users can select two-hop pre-requisites without unnecessary information.

Another student view will be selecting multiple courses for fulfilling the degree requirement. Each

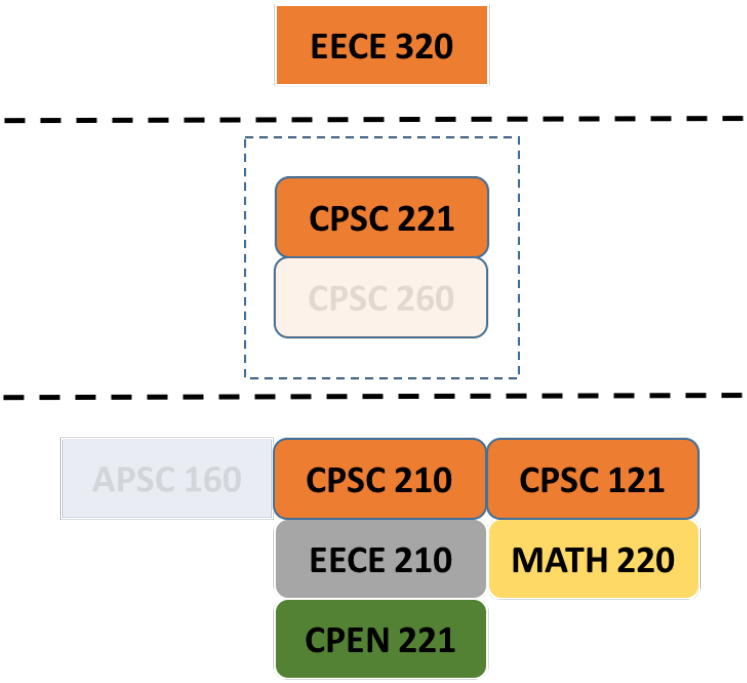

Figure 3: Example of removing irrelevant courses

course that students want to register might correlated with each other. For example, higher-level courses might have multiple options for the prerequisites, and higher-level courses in the same field might have some joint prerequisites. Thus, if the joint lower-level courses are useful for the student, it would be beneficial to register these joint prerequisites other than satisfying different prerequisites for those who had shared one. Besides, after choosing interested higher-level courses and determining the suitable prerequisites, total credits obtained from these courses can be calculated, which helps students to decide if these courses are enough for fulfilling the requirement for the degree. Figure 4 illustrates the case of joint lower-level course. For example, EECE 320 and CPSC 411 both requires CPSC 221 as the prerequisite. Besides, different color indicates courses provided by the different department.

2.2.2 Encoding for cor-requisites, equivalencies, and soft-wired prerequisites

The encoding for cor-requisites, equivalencies, and soft-wired prerequisites in student view is similar to the topology view. If two courses are cor-requisites, we can regard them as the mandatory prerequisite course to each other. For courses that are equivalent, we can use containers as the one of relationship. The visualization of soft-wired prerequisites is similar to topology view.

3 A scenario of Use

The topology view and the student view are jointly presented to users. By typing course names or clicking courses in the topology view, the visualization of these courses' prerequisite network will be generated. Once double clicked on a course, an option is given to user if she is interested in knowing whether or not she has satisfied the prerequisites for the course. If the answer is yes, the tool shall go to the student view. The user can then interact with the student view as mentioned in the previous section. The sketch of topology view is shown in Figure 1 and the sketches of student is shown in Figure 2-3.

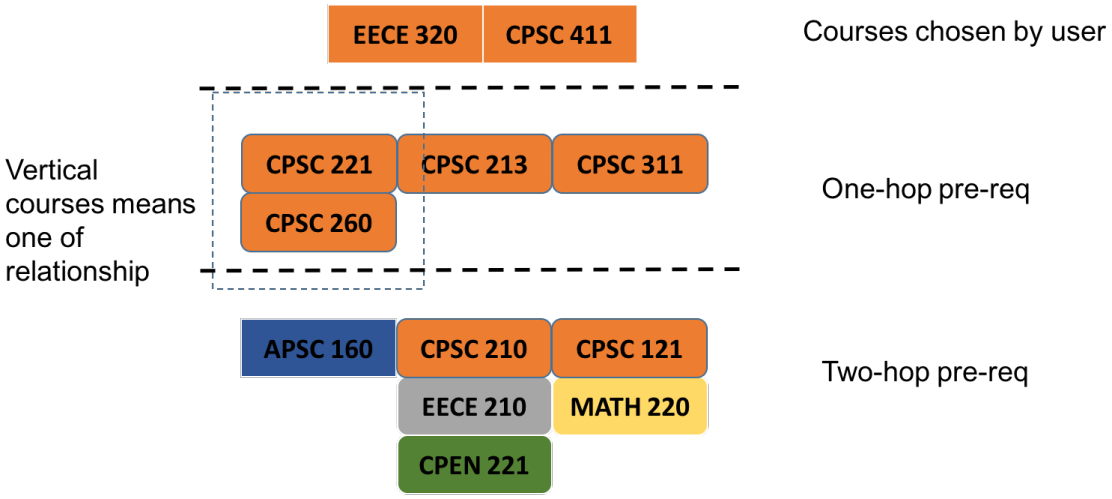

Figure 4: Example of Joint prerequisite visualization

4 Implementation approach

4.1 Data abstraction

The key to this visualization about understanding the prerequisite relationships among courses. We have downloaded a complete catelogue of UBC courses and extracted useful prerequisite information from it. By using simple regular expression methods, each course with their credits is obtained from this file. Then, the prerequisite requirements are also extracted. Typically, there are three types of prerequisites as mentioned earlier.

4.2 Data Structure of the Course Network

To store the information about prerequisites, we split all information into four tables: (a) course information table; (b) Mandatory prerequisite table; (c) *either* relationship prerequisite table; (d) *or* relationship prerequisite table. The course information table store basic course information, such as course name and the credits for the course. The course name can be used as the external key to indexing courses in other tables. Mandatory prerequisite table stores the information on mandatory prerequisites. This table has the attribute course name (key) and corresponding mandatory prerequisites. Then, *either* and *or* relationship tables have one more attribute than mandatory prerequisite table, which is an index of the either-or pair. This is because some courses might have multiple either-or pairs and multiple *or* courses of a *either* course. In this way, *either* course and *or* course(s) in the same either-or pair will have the same index, which makes this relationship determinate.

4.3 High-level implementation

We shall build our infovis solution on a website where frontend is primarily build on D3.js and Cytoscape.js. We will use Cytoscape.js for drawing containers. Currently, we plan to use D3.js for drawing student view but we may end up with using a library if that leaves us more time to work on the design choices of the visualization. Our tool support queries by having backend processing SQL commands and sending to the frontend before D3 and others can use it.

5 Milestones

Nov 07: Finish data abstraction and build up a database to store course prerequisite information.
Nov 14: Have a preliminary implementation of single course's prerequisites where we can view clearly

what are the selected course's one-hop prerequisites.
Nov 21: Finish multi-hop prerequisite for the single course.
Nov 28: Finish prerequisite courses for multiple selected course, which satisfied the requirement of scenarios two. student View completed.
Dec 05: Finish department/topology view for scenarios three. Finish liking of topology view to student view.
Dec 12: Presentation deadline.
Dec 15: Paper deadline.

6 Previous work

The CPN(Curriculum Prerequisite Network) [1] network visualization has been studied by Aldrich, where he focuses on the overall topology of the courses at Benedictine University. Aldrich proposed a DAG for the network, in particular, each edge represents a single relation between a pair of courses. It is surprising that the number of edges is smaller than the number of nodes in Aldrich's study. While the visualization indicates that math is the fundamental discipline of many disciplines and there exists some separation of arts and science, no further meaningful insights can be inferred by staring at the whole graph. In addition, the paper does not consider the *either or* prerequisite. There's also a work-in-progress project by Gestwicki [2] which follows the above CPN encoding idiom, but it is not known if the project has finished.

7 Expertise

Siyuan's Expertise: Siyuan has some theoretical experience of graph theory and networks, but he has no experience of automatically drawing graph data.
Jiahong's Expertise: Jiahong does not have much experience of creating visualizations system but do skilled in web programming.

8 bibliography

References

[1] Preston R Aldrich. The curriculum prerequisite network: a tool for visualizing and analyzing academic curricula. *arXiv preprint arXiv:1408.5340*, 2014.

[2] Paul Gestwicki. Work in progress-curriculum visualization. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, pages T3E–13. IEEE, 2008.