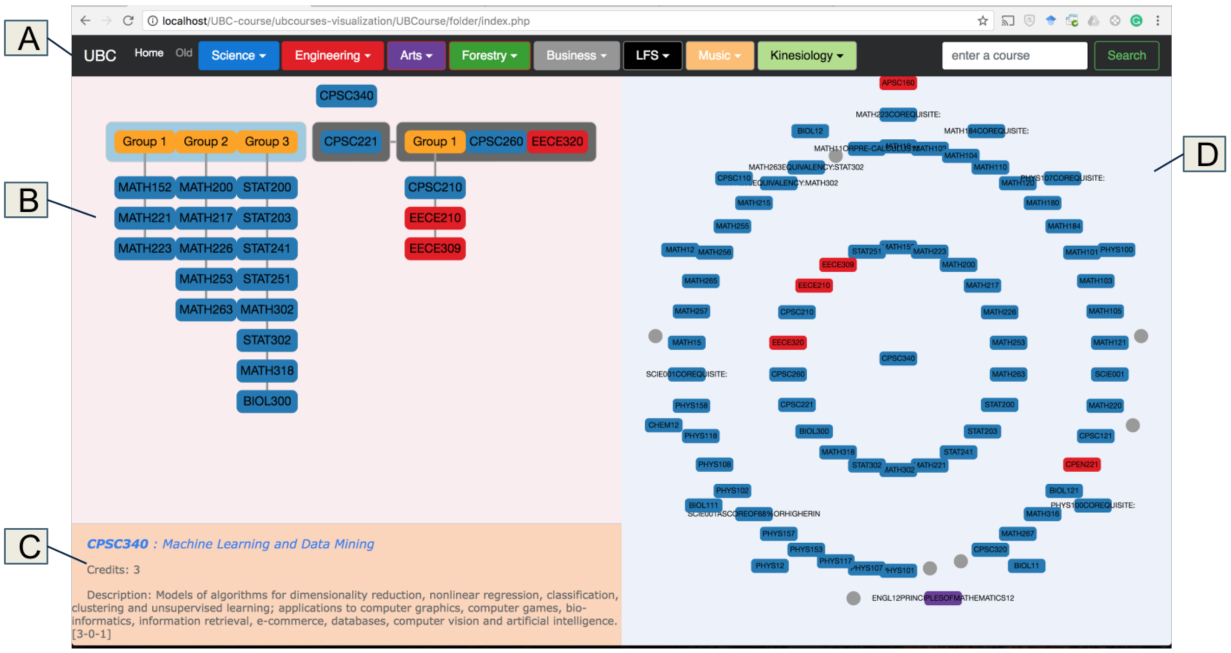
UBCourse Vis

Jiahong Chen, Siyuan He



CPSC340 : Machine Learning and Data Mining

Credits: 3

Description: Models of algorithms for dimensionality reduction, nonlinear regression, classification, clustering and unsupervised learning; applications to computer graphics, computer games, bio-informatics, information retrieval, e-commerce, databases, computer vision and artificial intelligence. [3-0-1]

Fig. 1. UBCourse Vis: Interface overview showing the course selection on the left and concentric view on the right.

Abstract—We present a novel course information visualization system. Aiming to help both students and curriculum planner, our system is designed with interactive aspects to best serve people with different needs. An interactive course selection scheme and an interactive department course catalogue viewer are introduced.

Index Terms—courses, visualization, networks

1 Introduction

The amount of data has grown exponentially since the last decade. In need of processing and consuming information, various visualization tool are developed for intuitive and insightful understanding. The course information from The University of British Columbia is no exception to the above rule. With more courses added each year and the apparent trend of multidisciplinary studies, courses offered by a department are taken by more students outside the department, and in consequence, courses have more and more prerequisites. In this project, we are interested in presenting the prerequisites relationships through visualization. Section 2 talks about the related work of our project. Section 3 and 4 details the problem in terms of the VAD framework [8]. Section 5 and section 6 talks about our solution and implementation. Section 7 provides use of scenarios of our system. Section 8 discusses the strengths and weaknesses of our work.

2 Related Work

Network visualization is usually drawn as node-link diagrams, where the vertex are represented as disks or boxes with a text label, and edges are shown in line segments [3,4]. The CPN(Curriculum Prerequisite Network) network visualization has been studied by Aldrich [2], where he focuses on the overall topology of the courses at Benedictine University. Aldrich proposed a directed acyclic graph for the network, in particular, each edge represents a single relation between a pair of courses. The width of edge directly encode the logical relationships such as $all\ of$ and $one\ of$. A work-in-progress project by Gestwicki [5] which follows the above CPN encoding idiom, but it is not known if the project has finished. However, the above papers does not consider the *either or* relations. This is accomplished in our project. Pienta and his colleagues designed an interactive visualization system for networked data, which helps people to explore the graph query results [9]. Zhao et al. developed BiDots to discovering and analyzing biclusters, i.e., two sets of related entities with close relationships [11]. Srinivasan et al proposed Orko, for facilitating multimodal interaction for visual exploration and analysis of networks [10]. Orko provides users with the freedom of expression via both natural language and touch-based direct manipulation input. Grtler et al. developed a circular treemap to intentionally allocate extra space for additional visual variables [6]. 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. During the course of the project, we also implemented a containment-based encoding (Figure 2), where containers with different hues signals different logical relations. After several rounds of peer reviews and discussions with Prof.Munzner, we find the encoding counter-intuitive and has few advantages over pure text descriptions.

- *Jiahong Chen E-mail: jhchen@mech.ubc.ca.*
- *Siyuan He E-mail: hesiyuan@cs.ubc.ca*

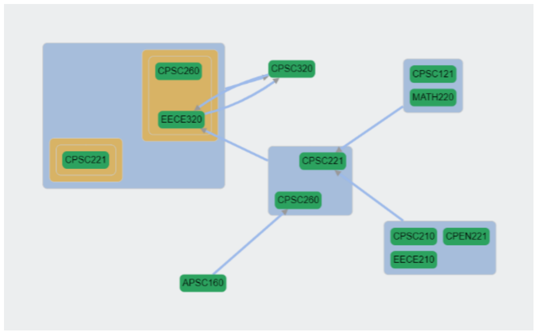


Fig. 2. Containment encoding are counter-intuitive

Number of course	Total relationship	Number of mandatory prerequisition	Number of either-or selections
7,521	8,381	6,931	1,460

Table 1. Summary of the Dataset

Field name	Description	Data type
course_key	The identifier of each course, 3-4 characters of department name plus 3-digit course ID	Varchar (10)
Number_credits	Credit of the course	int
course_name	Name of the course	Varchar(20)
course_description	Detailed course description as stated in the UBC Student Service Center	Varchar(300)

Table 2. Structure of credit table

Field name	Description	Data type
course_key	The identifier of each course, 3-4 characters of department name plus 3-digit course ID	Varchar (10)
group_details	details of the mandatory group	Varchar (300)

Table 3. Structure of mandatory group table

3 Data Abstractions

All information about UBC courses and theirs prerequisites are collected from UBC course schedule and UBC Student Service. A PDF file [1] containing information of courses such as the number of credits, their prerequisites, and short descriptions were downloaded and transformed into a plain text file. We then extracted course information and prerequisites relationship from this file. Table 1 summaries the dataset.

3.1 Data acquisition

Currently, the prerequisites of a course is formulated in the plain text of a semi-structured form. Thus, a great amount of data preprocessing should be done before data visualization. Prerequisites are captured as follow using Python:

- Regular expression is used to identify the start point of each course. Each course description starts with 3 to 4 character indicating department name, followed by a 3-digit course id. The number of credits of a given course is surrounded by a pair of parentheses.
- If a course has prerequisites before regular expression function matches next course name and credits, a paragraph including Prerequisition must appear. When courses have prerequisites, further processing will be taken to extract detailed prerequisites.
- Paragraphs indicating course prerequisites usually falls into two groups: *mandatory* and *either or*. All conditions stated in the mandatory group should be satisfied, and only one group within either-or groups needs to be satisfied. For example, if the prerequisite statement is "Prerequisite: A, either B or C or D", group A must be satisfied and only one of group B, C and D needs to be satisfied.
- Each group has several sentences about the detailed description of prerequisites. Normally, there are only two types of requirement, we ignore other requirements since the main topic of the course project is about data visualization, not natural language processing. The first type of requirement for the course is $all\ of$ requirement, which means all mentioned courses need to be taken. The second type of requirement is the $one\ of$ requirement, which means only one of those mentioned courses should be taken.

3.2 Database details

There are four tables stored in the MySQL database: course detail table, mandatory group table, either group table and or group table. Table 2 shows the structure of credit table, stores the credit of each course.

Table 3 indicates the structure of mandatory group table, which stores the information about mandatory prerequisites of courses.

Either group table and or group table are little different from the mandatory group. Several either group might be attached to only one either group, and there might be several either-or relationships in prerequisites of one course. Thus, they are not matched one by one, and an index should be stated to match each either-or pair. Table 4 shows the structure of either group table and or group table.

3.3 Data derivation

In addition to the above tabular data, we have some derived data.

- Adjacency List: Given some courses with the text description of their prerequisites, we build an adjacency list representation of their relationships.
- Weights: Given a course, we calculate the weight of each of the prerequisites.
- Hops: Given a course, we can recursively traverse the text description of prerequisites to finding all prerequisites including prerequisites of prerequisites of the course. The number of hops of each such prerequisite to reaching the course is derived and utilized in the concentric view.
- Number of prerequisites: Given a course, we can go through its prerequisites description to count the number of listed courses. This number reflects the complexity of description of prerequisites of the course.

We analyze the domain problem according to the VAD framework [8].

Field name	Description	Data type
course_key	The identifier of each course, 3-4 characters of department name plus 3-digit course ID	Varchar (10)
idx	Index of the either-or pair	int
group_details	details of the either/or group	Varchar (300)

Table 4. Structure of either group table and or group table

3.4 In abstract terms

From the above descriptions, our data in the abstract form is:

- Dataset Types: Text.
- Derived Dataset Types: Tables.
- Derived Derived Dataset Types: Adjacency List(Network).
- Data Types: Items, Attributes.
- Derived Data: Links, ordered attributes.
- DataSet Availability: Static.

4 Task Abstractions

4.1 In Domain Terms

These include but not limited to

- A student who wants to know if they satisfy all prerequisites for a course.
- A curriculum planner who wants to analyze dependencies of courses across departments.
- A student who wishes to plan their courses towards their degree.
- A student who loves to explore any graphs.
- A student who eagers to know how UBC works.

4.2 In Abstract Terms

We abstract the above task descriptions according to VAD book [8].

- Identify a node in the network
- Explore nodes in the network.
- Enjoy colorful nodes in the visualization.
- Discover node to node relationships.
- Record a network drawing.

5 Solution

UBCourse Vis aims to provide an interactive way to help students go through all prerequisites and make selections of them. Since not all prerequisites are needed, students can first select part of the prerequisites of the desired course (first-hop prerequisites) according to requirements. When the requirements of first-hop prerequisites are satisfied, second-hop pre-requites of the them will then pop out. By carrying out this process iteratively, multi-hop prerequisites of the desired course will be determined. Besides this, to help users understand the overview of the course prerequisites, concentric view and department view are provided.

To support tasks listed in Section 4, the main solutions provided by UBCourse Vis are as follows:

- Interactive prerequisite selection: exploring and selecting prerequisites of a single course hop-by-hop.
- Concentric view: A query-centric graph layout of a network.
- Department view: An interactive user-dependent grid layout of a network.

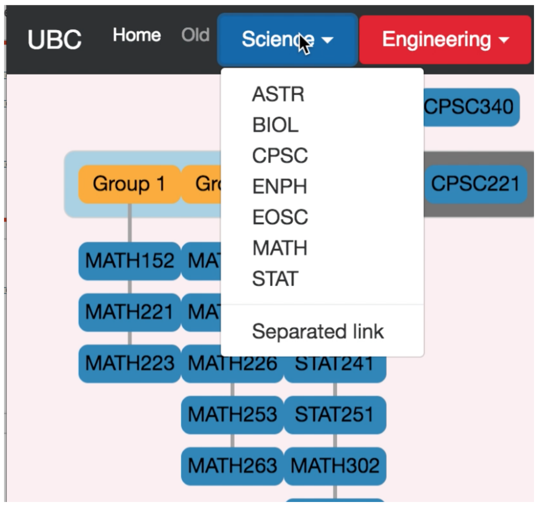


Fig. 3. Navigation bar drop down list.

5.1 UBCourse Interface Overview

The UBCourse Vis user interface contains four main areas as indicated in Figure 1. The Navigation bar (Figure. 1 (A)) on the top provides the information about the different faculties that provides undergraduate course and graduate courses in UBC. Each faculty has a drop-down list to navigate to different departments. By clicking each department, all courses provided by this department as well as their highly related course will be presented in a new window as Figure 7. Besides, we can also search the prerequisites of a specific course on the right end of the navigation bar. The web page will then jump to a new UBCourse interface showing detail about that course. Section 5.6 provides detailed information about this function. Then, the main window split into three sub-windows. Top-left window (Figure. 1 (B)) presents the interactive prerequisite selection for users. Users can select optional courses to satisfy all prerequisite requirements hop by hop. Bottom-left window (Figure. 1 (C)) give full detail of the queried course, including course name, credits, and the description of the course. Last, a concentric view (Figure. 1 (D)) illustrates the knowledge hierarchy from the queried course to its prerequisites. The VAD idiom used here is the Facet, which split the window into multiple parts.

5.2 Navigation bar

Since there are a significant number of the department offering a various course, it is meaningless, if not possible, to show all departments to users at one time. Thus, we summaries departments into different faculties and a drop-down list is provided with each faculty showing all related departments. As shown in Figure 3, departments relating to Science Faculty are summarized here.

Besides, we can also query prerequisites of a course by searching it over the box in the top right corner as shown in Figure. 1 (A). A new overview page will be generated with details about the new courses.

5.3 Interactive prerequisite selection

We designed an interactive prerequisite selection scheme to help students have a better understanding of the full course prerequisite relationship. This scheme will first present all one-hop prerequisites of the selected course. Different colors in the text box are used for encoding various faculties and colors in the container box are used for indicating mandatory group and optional groups. Node-link diagram is used for representing optional relationship. Besides, the horizontal spatial position of text boxes within the container box is used for encoding the mandatory selections.

Figure 4 represents the one-hop prerequisites of the course CPSC340. The plain text description for this course is:

- One of MATH 152, MATH 221, MATH 223 and one of MATH 200, MATH 217, MATH 226, MATH 253, MATH 263 and one of STAT 200, STAT 203, STAT 241, STAT 251, MATH 302, STAT 302, MATH 318, BIOL 300; and either (a) CPSC 221 or (b) all of CPSC 260, EECE 320 and one of CPSC 210, EECE 210, EECE 309.

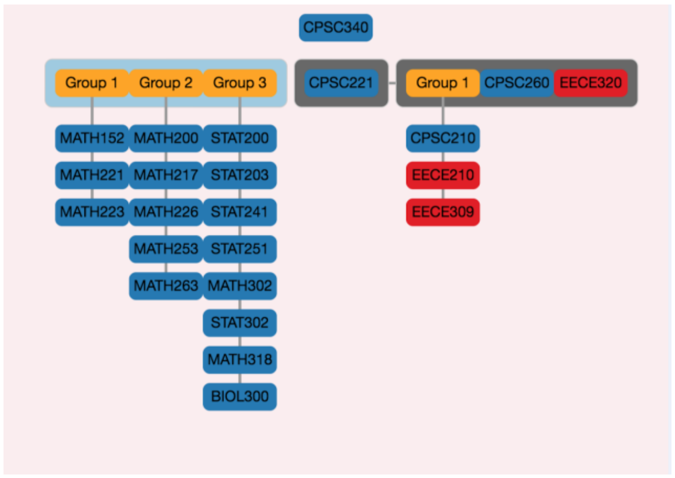


Fig. 4. One-hop prerequisites of CPSC340.

As shown in Figure 4, dark blue box and red box encoded for courses from Faculty of Science and Faculty of Applied Science (Engineering) respectively. Besides, light blue container box stands for the mandatory group of course, which means all conditions in this box should be satisfied. On the contrary, gray container box stands for *either or* groups; these optional boxes are connected via edges. Only one of these gray container box needs to be satisfied. For example, we only need to take CPSC 221 to satisfy all *either or* groups. As for orange boxes labeled with Group *x*, they are used for representing the 'one-of' relationships. Edges connect optional courses within the group, and vertical spatial position encodes this optional relationship. For example, the statement 'One of MATH 152, MATH 221, MATH 223' is encoded as the group 1. One of these three courses is needed to satisfy the requirement of this group.

After satisfying the prerequisites of the current course(s), next hop prerequisites will pop out automatically. To illustrate further hop actions, we select MATH 223, MATH 263 and BIOL 300 to satisfy the mandatory group and CPSC 221 to satisfy 'Either-or' group. The result is shown in Figure 5. Since not all lower-level undergraduate courses have prerequisites, the complexity of second-hop prerequisites is much less than the one-hop. Only two groups of mandatory course are required.

The VAD design idioms are used in this window: (a) Encode; (b) Reduce; (c) Navigate; (d) Select. Among them, three types of Encode method are used. Node-link Diagrams are used for presenting the network data; colors are used to encode different faculties and groups; spatial position are used to encode the relationship of mandatory and optional courses. Besides, data reduction is also introduced in this design study. Different courses might share same prerequisites. Thus, there are duplicated courses. And selecting such courses can satisfy multiple requirements. Such duplicated courses are automatically chosen to reduce the number of selection for users to take. Besides, already satisfied prerequisites will also be eliminated automatically. Moreover, the 'Zoom in/out' method of Navigation in data manipulation are also applied in the system. In case of some course have much too prerequisites, zoom in function can help students to know the detail of the prerequisite network. Last, students can select prerequisites based on their interest. After satisfying all the requirements, further hops of prerequisites will pop out if applicable.

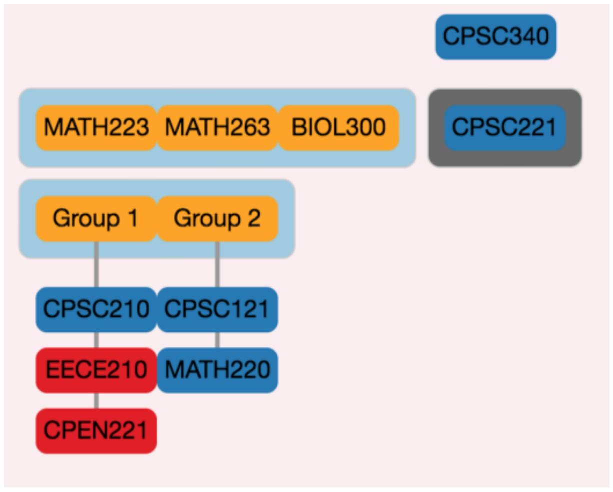
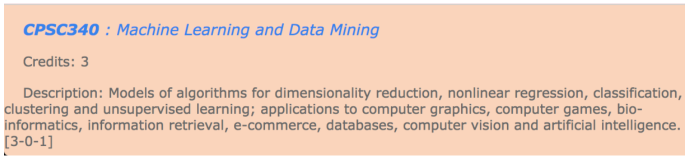


Fig. 5. Two-hop pre-requites with user interaction.

5.4 Course detail

In this window, course details are presented. Full course name and detailed course description will be presented to help students know more about the selected course. Due to the limitation of time, this window can only provide the information about the querying course. A 'Manipulation' design idiom should be added to change the course detail in this window when hovering over all listed courses. Such design will help student to learn more about the prerequisite courses.



CPSC340 *: Machine Learning and Data Mining*

Credits: 3

Description: Models of algorithms for dimensionality reduction, nonlinear regression, classification, clustering and unsupervised learning; applications to computer graphics, computer games, bio-informatics, information retrieval, e-commerce, databases, computer vision and artificial intelligence. [3-0-1]

Fig. 6. Navigation bar drop down list.

5.5 Concentric view

Given a query, we present a concentric view in addition to course selection on the right panel. With the queries course in the center, its one-hop prerequisites are equally aligned on a concentric circle surrounding the course. And the prerequisites of the prerequisites are equally placed on a concentric circle with a larger radius than the previous and so on. We do not simply draw all the edges at once since the overall hierarchy of these courses is of more importance than the particular relations within the network. A user may click on a particular course to show relations with its prerequisites. The relations are encoded directly by directed edges from courses to other courses. The relevant VAD idiom is *How* : *Encode* where we map derived data, the number of hops to a radial distance.

5.5.1 Edge Encoding

We use the width of edge to encode the derived attribute, the weight of each prerequisite of a course. The calculation method is as follows. The result is a weight vector for each course. This calculation method has some advantages such as fast performance, but indeed, it also has some limitations. Since the weights are used as the data to draw edges, it is not difficult for users to recognize the difference of widths among edges. Also, by the end of this calculation, we lose the logical information of the prerequisite description, and our encoding only approximates the original information. We address this issue further in the discussion section.

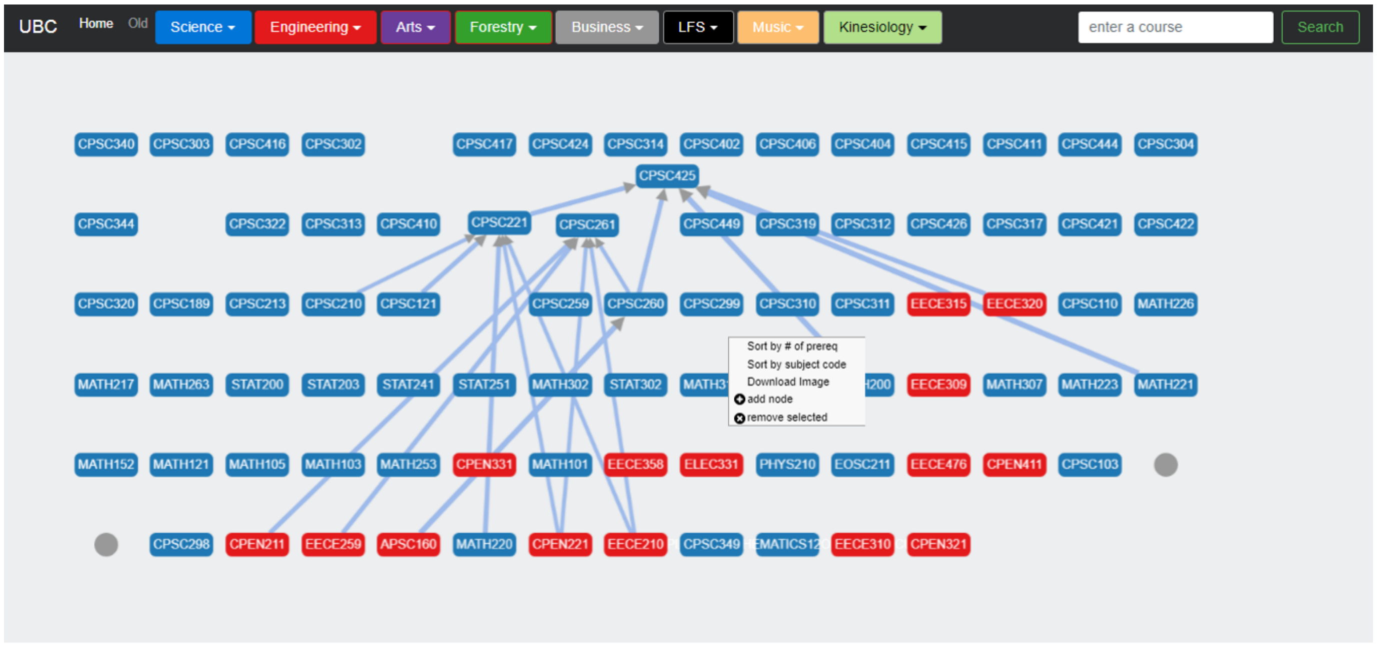

UBC
Home
Old
Science
Engineering
Arts
Forestry
Business
LFS
Music
Kinesiology
enter a course
Search
CPSC340 CPSC303 CPSC416 CPSC302 CPSC417 CPSC424 CPSC314 CPSC402 CPSC406 CPSC404 CPSC415 CPSC411 CPSC444 CPSC304
CPSC425
CPSC344 CPSC322 CPSC313 CPSC410 CPSC221 CPSC261 CPSC449 CPSC319 CPSC312 CPSC426 CPSC317 CPSC421 CPSC422
CPSC320 CPSC189 CPSC213 CPSC210 CPSC121 CPSC259 CPSC260 CPSC299 CPSC310 CPSC311 EECE315 EECE320 CPSC110 MATH226
MATH217 MATH263 STAT200 STAT203 STAT241 STAT251 MATH302 STAT302 MATH3 200 EECE309 MATH307 MATH223 MATH221
Sort by # of prereq
Sort by subject code
Download Image
add node
remove selected
MATH152 MATH121 MATH105 MATH103 MATH253 CPEN331 MATH101 EECE358 ELEC331 PHYS210 EOSC211 EECE476 CPEN411 CPSC103
CPSC298 CPEN211 EECE259 APSC160 MATH220 CPEN221 EECE210 CPSC349 EMATICS12 EECE310 CPEN321

Fig. 7. Computer Science Department

```
Data: Prerequisites of a course
Result: A weight vector of the course
for Course in all of groups do
    assign weight = 1
end
for course in one of group do
    assign weight = sizeof(group)
end
for course in either or group do
    inflate its weight by 2;
    process all of and one of similarly to above
end
Reassign each weight = max(weights) - weight;
```
Algorithm 1: Prerequisite Weight Calculation

5.6 Department view

The department view (Figure 7) is selected when a user clicks into a department in the drop-down lists. All courses offered by the department and courses that are in the prerequisites description of these courses are shown in a grid sorted horizontally and vertically by some heuristics. The default heuristic is the course subject code. We also offer users other heuristics such as the number of prerequisites. Due to the time constraint, we do not implement different possible layouts such as topological sort. Nevertheless, the above two heuristics approximately lays out the department courses vertically so that the directed edges are pointed upwards. As in the case of the concentric view, we do not draw the edges up front. Users may investigate some particular relations in the department. Thus we give users full control over the network. We discuss the trade-off of leaving the drawing to users in the section 8. The relevant encoding idioms are $How : Encode$ in which edge width encodes the derived data, the weight of each prerequisite respective to its parent, and $How : Manipulate$ where we can sort, select, move, add, and select based on user inputs.

6 Implementation

To create UBCourse Vis, several third-party libraries are used:

- **Cytoscape.js**: Cytoscape.js are used primarily to render the derived adjacency list into a graphic representation.
- **Cytoscape-context-menus.js**: An extension to Cytoscape.js that is used in the department view to support user menus.

Besides, several programming language and tools are used:

- **Python**: Python is used to process plain text data and extract useful UBC course prerequisite information. The processed data were put into CSV files to be suitable for loading into the database.
- **Google Chrome**: Page source viewer and the webpage inspector are utilized for debugging.
- **MySQL**: A web-based MySQL server is used for storing all the data processed by Python.
- **JavaScript**:Using Bootstrap, CSS, and HTML to coding the front-end of the webpage.
- **PHP**: PHP is used to transferring data from the back-end(database) to the front-end (JavaScript). PHP is also used extensively to convert text data into adjacency list form.

6.1 Work Breakdown

Jiahong worked on processing text data, setting up databases, creating interactive course selection. Siyuan worked on creating website interface, concentric view, and department view. Both of us have worked on transforming raw text data to a network-like data structure. The challenging part of the project is the fact that no structured data is readily available. After the text processing of the UBC course catalog, our data is still in text form. Hence a majority of our work lies in the backend processing where given a query, we transform the text data into an adjacency list and send the data to the frontend.

7 Scenario Walk Through

We provide scenarios for both students and department organizers.

7.1 Scenario for students

In this scenario, John, a first-year Computer Science student in UBC, decides to begin his university career as a Data Scientist. He is very keen on the applications of Machine and tries to enroll in CPSC 340, Machine Learning and Data Mining in his third year. The most complicated problem for him is that there more than 20 prerequisites for him to choose, and it is hard for him to know what course he should take in the first year. With the help of UBCourse Vis, he starts with searching prerequisite details, then interactively select interested prerequisite courses, using the information provided by all three windows as shown in Figure 4 (B-D), and ends by deciding all prerequisites of CPSC 340.

He then opened UBCourse Vis, typed the course in the search box. UBCourse Vis provides the overview for the prerequisite structure for him as shown in Figure 1. He then browses the directly linked course in the concentric view to find if he can take minimal prerequisites for CPSC 340. As shown in Figure 9, there seems to be a link from CPSC 340 to CPSC 221 then CPEN 221 and finally connects to APSC 160. Thus, he decides to select MATH 223, MATH 263, BIOL 300, and CPSC 221 as the one-hop prerequisites and CPEN221 and MATH 220 for two-hop prerequisites. Then, he finally gets the course that he needs to take in the first year, APSC 160. All selected prerequisites are then listed vertically as indicated in Figure 8.

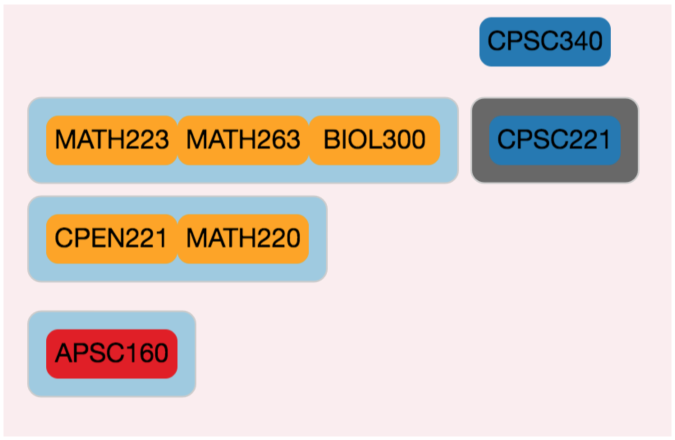


Fig. 8. All prerequisites after selection

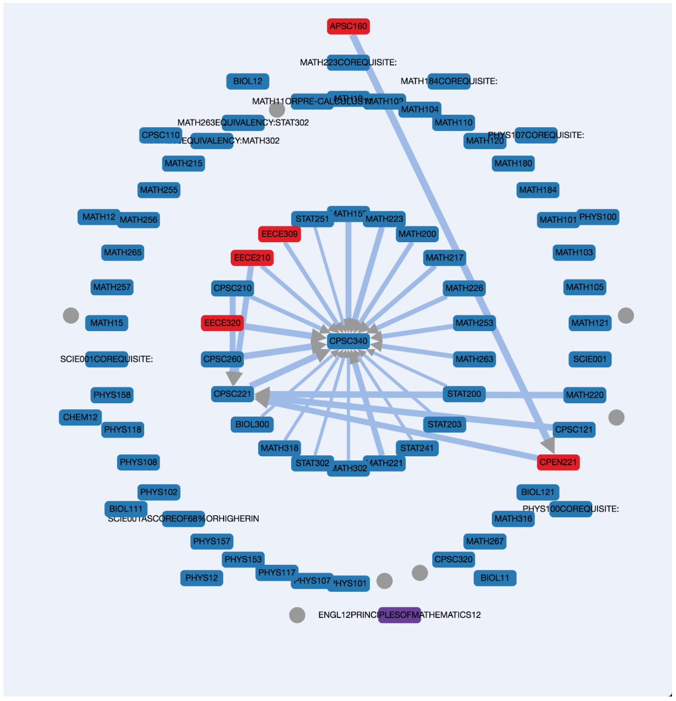


Fig. 9. Course link view for student

7.2 Scenario for department organizer

Steve is the CS department head who wants to reorganize the courses provided to undergraduate students. He tries to sort out the complex relationships among courses and see if there are any duplicated courses. He then navigates to department view as shown in Figure 3. Then he clicks into the CPSC. All CPSC courses and their prerequisites are displayed in the grid layout sorted by year level. He then right clicks on the page on select sorting by the number of prerequisites. All courses then transition to new places on the grid. He then clicks on CPSC 425, then all prerequisites of the course including CPSC 221 and EECE 320 are drawn with connection marks towards CPSC 425. He noticed that EECE320 is an old course, so he right clicks on that course and removes it. He then clicks on CPSC 221 to further showing the prerequisites of CPSC 221 as shown in Figure 7. After several rounds of clicking, moving and removing, he right-clicks on the page and selects download image. He is satisfied with his generated network and saves the image for a presentation of department change on the prerequisites of CPSC 425 that is coming in a few minutes.

8 Discussion and Future work

8.1 Encoding

Most prerequisites descriptions in UBC course catalogue [1] are free of complex logical expressions. However, theoretically, these expressions can become complicated as the number of course offering continue to increase. Thus it becomes difficult to encode local relations of some particular courses without compromising overall topology of the network. While the correctness of local relations is essential, the goal of the visualization is to understand the network from a high-level point of view better. Therefore, we decide only to approximate the local relations, that is, using directed edges with different widths. Whether encoding using different widths is useful, distinguishing edge widths sometimes is not an easy task. Even more, what does it mean for an edge to be encoded with width w? The visualization system does not explicitly mention the meaning of w. It is also possible to encode logical relations entirely with the introduction of logical operator nodes, where a node in the graph is either a course node or a logical node. A course node is only connected with a set of logical nodes. We leave the encoding scheme as a future work for other people. Initially, we start with the project in the hope of encoding relations accurately and we end up with the hue-coded containers (Figure 2), and the implementation of containers is technically difficult. However, technically difficult does not mean user-friendly. It is more helpful to talk to a few people and extract some use cases from them rather than diving into the technical part at first.

8.2 Layout

As far as UBCourse vis is concerned, it generates three different layouts, vertical layout, concentric layout and grid layout. The vertical layout in the interactive course selection panel and the concentric layout serves the same purpose though they look different. Using vertical layout strongly indicates the year level of the courses, or order of course sequences. Arguably, it can happen that courses may not have space to fit on the screen since there are more courses in each layer as we are going from top to the bottom. The concentric view certainly illustrates this point. However, a user usually clicks on a subset of courses during the interaction and so not all courses on relevant to the user thus we would not have many courses showing on the next layer. The concentric view gives the big picture of the query course about all its prerequisites. It is better than using the vertical layout in the case by the previous argument. The grid layout is sorted using different heuristics to approximate the result of the topological sort and at the same time display all courses equally at once. We believe the vertical layout and the concentric layout are easy to understand, but as for the grid layout, it does not convey any strong information of the data since it equally displaced the courses on the screen. The remedy of sorting heuristics are helpful but still do not make the big picture standing out. If we had more time, we would implement a topological sort layout algorithm.

8.3 Interaction

UBCourse vis heavily relies on the interaction to be useful. The course selection view is purely interactive and based on the input of users, it generates outputs accordingly. This scheme efficiently reduces the number of courses showing on each layer. However, it may be helpful to explain additional instructions in the form of tool-tips so that users know what they are doing. The concentric view and department view share the same interaction part in that relations are only drawn upon user clicking. This interaction scheme is relevant in the concentric view since the layout conveys a big picture directly and we would imagine all edges are going from outside to inside. The edges are only necessary if some relations are essential, but it is a big problem that we do not know what users are specifically interested. This again made us reflect on the importance of refining use cases first before going towards the design and implementation. Some additional interactions are introduced in the department view such as sorting courses, removing courses, and adding courses. It becomes increasingly clear that we should also not display all the courses at the start. Rather, displaying a few important courses calculated by some heuristics such as node degrees, and then showing other courses on user demand could be a more effective interactive scheme. The idea of showing nodes on demand can be found in the interactive visualization of genealogical graphs [7].

9 Conclusions

We present an interactive visualization system of UBC courses. For querying on a particular course entry, we introduce an interactive course selection scheme in conjunction with a concentric view showing the knowledge hierarchy. For browsing the course catalogue of a department, we present a flexible grid layout of department courses that is capable both showing overview information and also detailed relations.

Acknowledgments

The authors wish to thank Prof.Munzer and peer reviewers in CPSC 547 for their feedbacks. This work is supported by UBC.

References

[1] Ubc vancouver calendar courses. http://www.calendar.ubc.ca/vancouver/pdf/UBC$_{V}ancouver_{C}alendar_{C}ourses.pdf.Accessed:$ $2017-12-15$.

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

[3] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph drawing: algorithms for the visualization of graphs*. Prentice Hall PTR, 1998.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

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

[6] J. Görtler, C. Schulz, D. Weiskopf, and O. Deussen. Bubble treemaps for uncertainty visualization. *IEEE transactions on visualization and computer graphics*, 2017.

[7] M. J. McGuffin and R. Balakrishnan. Interactive visualization of genealogical graphs. In *Proceedings of IEEE Symposium on Information Visualization (InfoVis) 2005*, pp. 17–24, October 2005.

[8] T. Munzner. Visualization analysis and design. *New York: A K Peters/CRC Press*, 2014.

[9] R. Pienta, F. Hohman, A. Endert, A. Tamersoy, K. Roundy, C. Gates, S. Navathe, and D. H. Chau. Vigor: Interactive visual exploration of graph query results. *IEEE transactions on visualization and computer graphics*, 2017.

[10] A. Srinivasan and J. Stasko. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE transactions on visualization and computer graphics*, 2017.

[11] J. Zhao, M. Sun, F. Chen, and P. Chiu. Bidots: Visual exploration of weighted biclusters. *IEEE transactions on visualization and computer graphics*, 2017.