On Competition for Undergraduate Co-op Placements: A Graph Mining Approach

Yuheng Jiang and Lukasz Golab
University of Waterloo, Canada
{y29jiang,lgolab}@uwaterloo.ca

ABSTRACT

We propose a graph mining methodology to analyze the relationships among academic programs from the point of view of co-operative education. The input consists of student - job interview pairs, with each student labelled with his or her academic program. From this input, we build a weighted directed graph, which we refer to as a program graph, in which vertices correspond to academic programs and edge weights denote the percentage of jobs that interviewed at least one student from both programs. We show that various properties of this graph have natural interpretations in terms of the relationships among academic programs and competition for co-op jobs. We also present a case study that illustrates the utility of the proposed methodology.

1. INTRODUCTION

According to the World Association for Cooperative and Work-integrated Education, 275 institutions from 37 countries have implemented cooperative education (co-op) programs [17]. Co-op experiences are vital because they supplement students' classroom skills and help them to gain practical experience.

We propose a graph mining methodology to analyze the relationships and competition among academic programs in the context of co-op. Our motivation is threefold. First, with academic institutions introducing new programs in recent years [6, 15], it is often unclear how one program differs from another. As a result, employers may not know which programs to advertise their jobs to and students may not realize that they qualify for a job targeted to a related program (e.g., Computer Science vs. Software Engineering). Understanding similarities among programs can lead to more effective job and academic classification schemes and therefore can help match job opportunities with qualified students. This analysis can also help students choose programs of study that correspond to their desired careers. Second, data from the co-op system may be used to identify multi-disciplinary programs that enable their students to obtain various types of jobs. This issue is becoming increasingly important given the recent rise in popularity of multi-disciplinary and well-rounded education [1, 2, 5, 10, 16]. Third, analyzing co-op job data can reveal jobs that are exclusive to particular departments, and, conversely, departments whose students compete for jobs with students from other departments. The university can choose to attract more employers that offer jobs to programs facing strong competition. Thus, the problems we study in this paper are critical to co-operative education from the student's, employer's and institution's perspective.

While some of these questions have been raised in prior work (details in Section 2), we propose a data-driven technique for answering them. Our input consists of student - job interview pairs, with each student labelled with his or her academic program. We transform this input to a graph, which we refer to as a *program graph*, in which vertices correspond to academic programs and edge weights denote the percentage of jobs that interviewed at least one student from both programs. Thus, the larger the edge weight, the stronger the relationship and competition between two programs.

Within the program graph, we are interested in vertices forming clusters or communities, vertices that are connected to many such clusters, and vertices that are strongly connected to their neighbours. As we will show, these graph properties have natural interpretations in the context of co-op. Graph clustering and community detection determine groups of related programs whose students interview for the same types of jobs; programs with connections to multiple clusters are likely to be multi-disciplinary; and programs with strong connections to their immediate neighbours face strong competition for jobs.

2. RELATED WORK

The majority of related work qualitatively or statistically analyzed co-op education through survey data with fewer than 100 entries. To the best of our knowledge, the first research work that used a large-scale data-driven methodology was our previous work [9]. We analyzed satisfaction with the co-op process using three years of evaluation data (students' evaluations of their employers and employers' evaluations of students). We found that students received better evaluations in their senior years, but they rated their first employer the highest. We also found that senior students outperformed junior students in work placements abroad, and extended work terms at the same employer (spanning more than one academic term) did not increase student satisfaction. In this paper, we target a different problem of understanding the relationships among academic programs.

In the context of academic programs, Wilson and other researchers urged traditional academic disciplines to be updated to better reflect reality [6, 15]. Furthermore, Hesketh found that employers have trouble advertising to specific programs and instead they advertise based on desired skillsets [8]. As we will show, clusters in the program graph indicate similar programs and suggest related programs that employers can advertise their jobs to. Additionally, it was suggested that programs can be evaluated based on their students' ability to obtain jobs [7, 14], which is a question that can be answered with the help of our methodology. Also, while the importance of multi-disciplinary education has been widely recognized [1, 2, 5, 10, 16], we propose a data-driven methodology for analyzing whether students from a particular academic program qualify for different types of jobs.

3. METHODOLOGY

We are given a dataset corresponding to student - job interview pairs, with each student labeled with his or her academic program and each interview associated with a job ID. We propose a methodology that relies on transforming the student-job interview pairs to an edge-weighted directed graph $G = (V, E)$, with a set of vertices V and a set of edges E. Vertices correspond to academic programs and edges represent relationships among programs. Let e_{ij} be the weight of the edge E_{ij} from vertex v_i to v_j, and let J_i be the list of distinct jobs that interviewed students from program v_i. We define e_{ij} as the fraction of jobs that interviewed at least one student from both programs; i.e., the fraction of jobs in J_i that also appear in J_j:

$$e_{ij} = \frac{|J_i \cap J_j|}{|J_i|} \quad (1)$$

This can also be interpreted as a conditional probability that a job interviewed at least one student from program v_j given that it interviewed at least one student from program v_i.

The direction of edges is important. For a program node v_i, an incoming edge weight from v_j measures the fraction of jobs in J_j that also interviewed at least one student from v_i. Thus, a large incoming edge weight of v_i from v_j means that most jobs interviewing at least one student from v_j also interviewed at least one student from v_i. Conversely, a large outgoing edge weight from v_i to v_j means that most jobs interviewing at least one student from v_i also interviewed at least one student from the other program.

We give an example in Table 1, which corresponds to 4 jobs, 9 interviews and 8 students from three programs (A, B and C). The job lists for each program are: $J_A = \{1, 2, 3\}$, $J_B = \{1, 2\}$, and $J_C = \{2, 4\}$. The corresponding program graph is shown in Figure 1, and the edges are colour-coded by the source vertex. The edge weight from Program A to Program B is $|\{1, 2\}|/|\{1, 2, 3\}| = 2/3 = 0.67$, meaning that 67 percent of jobs that interviewed at least one student from Program A also interviewed at least one student from Program B. The edge weight from Program B to Program A is $|\{1, 2\}|/|\{1, 2\}| = 2/2 = 1$, meaning that every job which interviewed a student from program B also interviewed a student from program A. Thus, the larger the edge weight, the stronger the relationship and competition between two programs.

Our definition of edge weights assumes that a relationship between two programs exists if at least one student from both programs *interviewed* for the same job; if there are many such jobs, then the edge weight will be larger.

Having explained how the program graph is constructed, we now clarify how properties of the program graph are related to the types and extent of relationships among academic programs in the context of co-op jobs:

Table 1: Sample interview data

Student ID	Program Name	Job ID
1	A	1
2	C	2
3	B	1
3	B	2
4	B	1
5	A	2
6	A	3
7	C	2
8	C	4

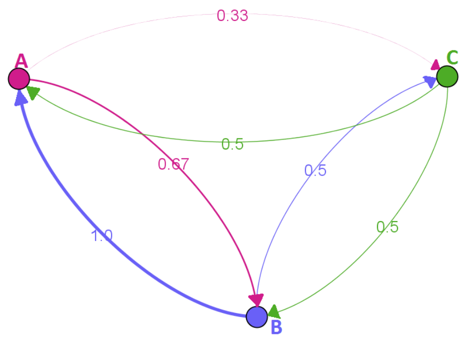


Figure 1: An example of a program graph

- **Clusters:** Clusters in a graph represent closely connected vertices. In our context, clusters represent related programs whose students interview for (mostly) the same jobs.
- **Outliers:** Given a graph clustering, we define outliers as vertices that have strong connections to other vertices from multiple clusters (as opposed to "normal" vertices connected mostly to other vertices within the same cluster). In our analysis, outliers correspond to multi-disciplinary programs: students from those programs have interviews in common with students from several different program clusters.
- **Fan-out:** (Weighted) fan-out measures the (weighted) number of outgoing edges of a vertex. In our context, weighted fan-out corresponds to the competition that a program faces from other programs. High weighted fan-out means that most jobs interviewing at least one student from the given program also interviewed students from other programs. As we will explain shortly, we use a modified version of standard weighted fan-out that takes into account the fact that our edge weights are defined in terms of set intersections (of the job sets of different programs).

In the remainder of this section, we describe the graph algorithms that may be used to identify program clusters, multi-disciplinary programs and programs facing strong competition.

3.1 Finding Clusters of Similar Programs

We use two techniques to find clusters of similar programs: near-clique finding and community detection.

The density of a graph (or subgraph) is the number of edges divided by the maximum possible number of edges, i.e., $\frac{|E|}{|V|*(|V|-1)}$. A clique is a group of vertices that are fully connected and therefore have a density of one. A near-clique is a group of vertices where the subgraph consisting of them and their edges has a density of nearly one, i.e., a group of vertices that is nearly fully connected. However, since our program graph is weighted and directed, we want to find near-cliques with large edge weights. To do this, we first remove all edges from the program graph except the five percent with the largest edge weights. The resulting graph may leave some vertices disconnected, while other pairs of vertices may only have an incoming or an outgoing edge. Then, we remove edge directions and simply retain an edge between two programs if there is either an incoming or an outgoing edge. Finally, we return all near-cliques from the resulting graph with density of at least 0.8.

In addition to identifying densely connected subgraphs via near-clique finding, we use the Louvain Modularity algorithm [4] to partition the vertices into disjoint clusters (communities), such that vertices with the same cluster are densely connected and vertices in different clusters are sparsely connected. This algorithm is included in many graph mining tools such as Gephi [3] and aims to maximize *modularity*, which compares the sum of the weights of intra-cluster edges resulting from given clustering with that of a randomly connected graph with the same number of edges [13].

Newman [12] introduced modularity for weighted undirected graphs. We translate this metric to weighted directed graphs as follows. Let c_i be the community that a vertex v_i belongs to, and $m = \sum_{ij} e_{ij}$, i.e., the sum of all the edge weights in the graph. The fraction of the edge weights that are intra-cluster is $\frac{1}{m}\sum_j e_{ij}\delta(c_i, c_j)$, where $\delta(c_i, c_j)$ is equal to 1 if $c_i = c_j$ (i.e. vertices v_i and v_j belong to the same cluster) and 0 otherwise.

Let $k_i = \sum_j e_{ij}$ (i.e., the sum of the weights of the edges that connect to vertex v_i). Consider another graph in which the fan-outs of all the vertices are the same but the edges are randomly connected. In such a graph, the probability of an edge existing between vertices v_i and v_j is $\frac{k_i k_j}{2m}$. The modularity of a graph clustering is defined as:

$$Q = \frac{1}{m}\sum_{i,j}(e_{ij} - \frac{k_i k_j}{m})\delta(c_i, c_j) \qquad (2)$$

$Q = 0$ means that the community detection result is no better than random. The maximum value for Q is 1. Higher modularity indicates more effective partitioning with more intra-cluster edges and fewer inter-cluster edges.

The Louvain Modularity method is iterative and includes two phases. In the first phase, each vertex starts in a different community. Then, for each vertex v_i, we compute the gain in modularity if v_i is moved to the community that its neighbour (v_j) belongs to. If the gain is positive, the change happens; otherwise v_i remains in its original community. This process is repeated iteratively and sequentially until no further improvements can be made. The outcome of the first phase is only a local optimum of modularity since the order of processing of the vertices will affect the result. In the second phase, a new graph is created such that the vertices are the communities obtained in the first phase, and edge weights are the sums of edge weights between vertices in the two communities. We reapply the process in the first phase on this new graph. The algorithm stops when maximum modularity is reached. To account for the effect of order, we run this algorithm multiple times and keep the result with the highest modularity.

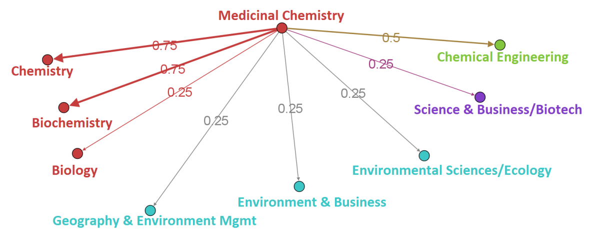


Figure 2: Direct competitors of Medicinal Chemistry, colour-coded by clusters

One characteristic of this algorithm is that it avoids creating small clusters. Lambiotte et al. [11] add a *resolution parameter* t to control the number of clusters. The new modularity definition is shown in Equation 3. The default t value is 1; smaller values of t lead to more and smaller communities.

$$Q_{new}(t) = (1-t) + \frac{1}{m}\sum_{i,j}(e_{ij}t - \frac{k_i k_j}{m})\delta(c_i, c_j) \qquad (3)$$

3.2 Finding Multi-Disciplinary Programs

To find multi-disciplinary programs, we start with the clusters/communities obtained by the Louvain Modularity algorithm. Intuitively, if an academic program has strong connections to other programs from multiple clusters (each of which corresponds to different types of jobs), it may be multi-disciplinary.

For each program, we propose a multi-disciplinary score as follows. For each cluster c_i identified by the Louvain Modularity algorithm, let p_i be the fraction of the total weight of the outgoing edges from the given program to the programs only in c_i. Then, for a given program, we compute the entropy of the distribution of edge weights among different communities simply as $\sum_i -p_i \log_2 p_i$. High entropy means that the given program has strong links to programs in multiple clusters and therefore may be multi-disciplinary.

We illustrate this concept with an example. Suppose that students in the Medicinal Chemistry program had interviews in common with students from eight other programs belonging to four clusters, labeled red, blue, purple, and green, as shown in Figure 2, with vertices colour-coded by their clusters. Only the outgoing edges from Medicinal Chemistry are relevant since they represent the percentage of jobs from $J_{MedicinalChemistry}$ that also interviews students from its neighbour programs. The sum of all out-going edge weights of Medicinal Chemistry is 3.25. $p_{red} = (\sum_{i \in \text{red cluster}} e_{MedicinalChemistry,i})/3.25 = (0.75 + 0.75 + 0.25)/3.25 = 0.54$, which is the sum of weights of edges from Medicinal Chemistry to the programs in the red cluster. Similarly, $p_{blue} = 0.23$, $p_{green} = 0.15$, and $p_{purple} = 0.08$. Thus, the multidisciplinary score of Medicinal Chemistry is $-p_{red}\log_2 p_{red} - p_{blue}\log_2 p_{blue} - p_{purple}\log_2 p_{purple} - p_{green}\log_2 p_{green} = 1.67$.

3.3 Finding Programs Facing Competition

We define the extent of competition that a program faces using a "set fan-out" metric. We want to compute the fraction of jobs that interviewed students from the given program which also interviewed at least one student from another program. For a given vertex (program) v_i, we define:

$$\text{Set Fan Out}_i = \frac{|\cup_{j \neq i} (J_i \cap J_j)|}{|J_i|} \quad (4)$$

A set fan-out of zero means that all the jobs that interviewed at least one student from program v_i only interviewed students from v_i and no other program. Students from such a program may have specialized skills that students from other programs do not have. A set fan-out of one means that every job that interviewed at least one student from program v_i also interviewed at least one student from another program. In other words, there were no jobs that exclusively interviewed students from v_i and therefore students from v_i may be facing strong competition for jobs.

Returning to Table 1, $J_A = \{1,2,3\}$, $J_B = \{1,2\}$, and $J_C = \{2,4\}$. For Program A, its set fan-out is $\frac{|(J_A \cap J_B) \cup (J_A \cap J_C)|}{|J_A|} = \frac{|\{1,2\}|}{|\{1,2,3\}|} = \frac{2}{3} = 0.67$. It means that students from Program A competed with students from other programs in 67 percent of their jobs. 33 percent of jobs that interviewed students from Program A did not interview students from other programs. The set fan-out for Program B is 1 and for Program C it is 0.5.

4. CASE STUDY

We now describe a case study that illustrates the utility of the proposed methodology. To carry out the analysis, we used the Gephi toolkit [3] which includes the Louvain Modularity algorithm. We used data from a large Canadian university including all interviews taking place in summer 2014, for co-op jobs taking place in Fall 2014. For each student - interview pair, the dataset includes the student's academic program and year, and job information such as the company name, job title, and targeted programs and academic years. The dataset consists of 4,194 students from 93 academic programs, 2,890 jobs and 16,855 interviews. On average, each job interviewed 5.8 students and each student had 4 interviews.

This academic institution has six faculties, each comprised of a number of academic programs: Science (programs include Physics and Earth Sciences), Mathematics (programs include Computer Science and Actuarial Science), Engineering (programs include Electrical, Mechanical, Civil, etc.), Arts (programs include Economics, Psychology and Sociology), Environment (programs include Planning and Geomatics) and Applied Health Science (AHS) (programs include Kinesiology and Recreation and Leisure Studies). All Engineering programs and several programs from other faculties (mainly Mathematics) have mandatory co-op education; other programs have optional co-op. As a result, most of the students and jobs in our dataset are from Engineering and Mathematics.

Rather than using all available data, we build the program graph using only the interviews of *senior* students (in their third and fourth academic years). Junior-level jobs tend to be less specialized, meaning that (junior) students from many different departments may qualify for an interview. In particular, we noticed that entry-level computer programming jobs interview students from many programs, including those outside computing. By focusing on senior students, we avoid generating edges in the program graph that correspond to junior-level jobs and may not truly indicate a relationship between programs. The resulting program graph contains 88 vertices (corresponding to programs that have at least two senior students in co-op) and 1,315 pairs of directed edges.

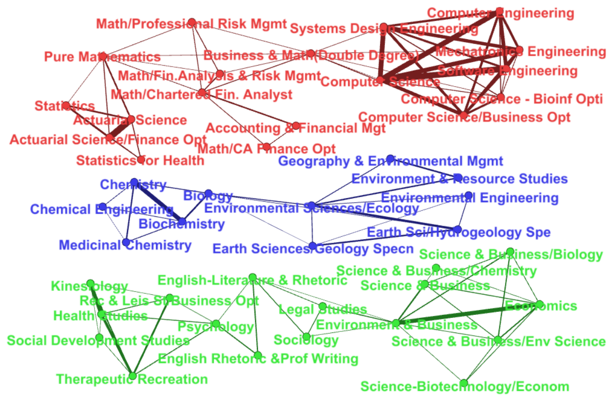

Math/Professional Risk Mgmt
Systems Design Engineering
Computer Engineering
Pure Mathematics
Business & Math(Double Degree)
Mechatronics Engineering
Math/Fin.Analysis & Risk Mgmt
Software Engineering
Computer Science
Math/Chartered Fin. Analyst
Computer Science - Bioinf Opti
Statistics
Computer Science/Business Opt
Actuarial Science
Accounting & Financial Mgt
Actuarial Science/Finance Opt
Math/CA Finance Opt
Statistics for Health
Geography & Environmental Mgmt
Environment & Resource Studies
Chemistry
Environmental Engineering
Biology
Chemical Engineering
Environmental Sciences/Ecology
Biochemistry
Earth Sci/Hydrogeology Spe
Medicinal Chemistry
Earth Sciences/Geology Specn
Science & Business/Biology
Science & Business/Chemistry
English-Literature & Rhetoric
Kinesiology
Science & Business
Rec & Leis St/Business Opt
Economics
Legal Studies
Health Studies
Psychology
Environment & Business
Social Development Studies
Sociology
Science & Business/Env Science
English Rhetoric &Prof Writing
Therapeutic Recreation
Science-Biotechnology/Econom

Figure 3: Vertices and edges participating in near-cliques

The program graph is a single connected component, i.e., there exists a path from every vertex to another. Its density is 0.34, meaning that one third of all possible program pairs had at least one interview in common. On average, the length of the shortest path between any two vertices is 1.7 and the diameter of the graph (i.e., the maximum length of any shortest path between two vertices) is three. The number of edges per vertex ranges from 4 to 66, with an average of 30.

4.1 Finding Clusters of Similar Programs

4.1.1 Near-Clique Finding

We begin by identifying near-cliques in the program graph (but considering only the five percent of edges with the largest weights, as described in Section 3). Figure 3 plots a subgraph of the program graph containing only the 46 vertices and 104 edges (in the top 5 percent of edge weights) that participate in the 25 near-cliques that we found. Three groups of programs appear to participate in the near-cliques, and we use a different colour for each. The larger the edge weight, the thicker the edge.

The red group at the top contains programs related to computing and maths. There is one near-clique with Software Engineering, Computer Engineering, Computer Science, Systems Design Engineering and Mechatronics Engineering. This suggests that Systems Design and Mechatronics students compete (interview) for software and programming jobs with students from core computing programs such as Computer Science. There are also two smaller near-cliques corresponding to Statistics/Actuarial Science and Accounting/Financial Analysis. Additionally, Pure Mathematics is connected to both of these; in fact Pure Mathematics students had interviews in common with students from 18 other programs. This suggests that Pure Mathematics students also interview for jobs in statistics, finance and business. Upon further inspection, we found that most such jobs were in financial trading.

The blue group of vertices in the middle includes two near-cliques: one with Chemistry-related programs and one with Earth Science and Environment-related programs. Based on these observations, the university may choose to either merge some of these related programs or redesign them to remove some of the overlap.

The green group at the bottom shows interesting connections. For instance, Economics seems strongly connected to Science & Business and Environment & Business, suggesting that these joint programs focus more on business than science (otherwise they would be connected with programs such as Chemistry and Environmental Engineering). Furthermore, there is a near-clique with seemingly unrelated programs: Sociology, Legal Studies, English-Literature & Rhetoric and Environment & Business; the first three are in the faculty of Arts while the last one is in the faculty of Environment. Upon further inspection, we found that the jobs these programs competed for were mainly in marketing and communications.

4.1.2 *Community Detection*

Next, we run the Louvain Modularity algorithm with different values of the resolution parameter to obtain a partitioning of the vertices into different numbers of communities, from 2 to 7. For example, Figure 4 shows the 7 communities we found, with each community in a different colour. For readability, we only include the edges in the top 5 percent of largest weights. Notice that Figure 3 is a subgraph of Figure 4, so all the near-cliques identified there are also visible here.

With these seven clusters, we obtain a partitioning into Engineering/Computing, Math/Finance, Natural Sciences, Social Sciences, Science &Business, Environment, and Health Sciences. Note that some engineering programs such as Chemical are placed in the Natural Sciences cluster and others such as Civil and Geological are placed in the Environment cluster. With only four clusters (illustration omitted for brevity), we obtain Engineering/Computing, Math/Finance, Natural Science/Environment, and Social/Health Science. With only two clusters (illustration omitted for brevity), we distinguish between Math/Engineering and Natural/Social Science programs.

4.2 Finding Multi-Disciplinary Programs

Recall that our methodology for identifying multi-disciplinary programs requires a clustering; then, for each program, we compute the entropy of its edge weight distribution across different clusters. We use the seven clusters from Figure 4 and obtained entropy values between 0.64 and 1.89. The top five multi-disciplinary programs (highest entropy) are: Science & Business/Biochemistry, English Literary Studies, Science & Business/Environmental Science, Biology, and Science & Business. The top five least multi-disciplinary programs are: Geological Engineering, Software Engineering, French, Mechatronics Engineering and Civil Engineering. Not surprisingly, joint programs of the form Science & Business were identified as multi-disciplinary while specialized engineering programs were not.

4.3 Finding Programs Facing Competition

We now search for programs with high set fan-out, i.e., those with few jobs that interviewed students only from that particular program. We found that for about half the programs, over 90 percent of the jobs that interviewed a student from a particular program also interviewed at least one student from another program. Thus, competition for jobs among academic programs appears relatively high. In particular, 16 programs, including Business & Mathematics, did not have any jobs that interviewed only their students (the jobs for which these students interviewed were computing-related or financial). Most of these 16 programs were small (only 3-4 senior co-op students). There were few jobs that specifically target these programs, so students from these programs had to interview for jobs advertised to other programs.

On the other hand, there were 8 programs where more than 30 percent of the jobs that interviewed at least one of their students did not

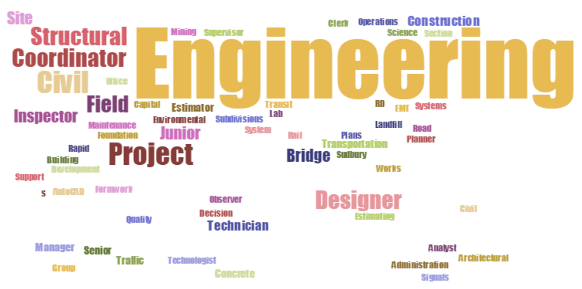


Figure 5: Word cloud of job titles of 70 Civil Engineering jobs that only interviewed students from Civil Engineering

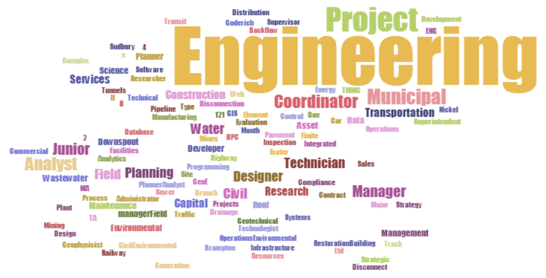
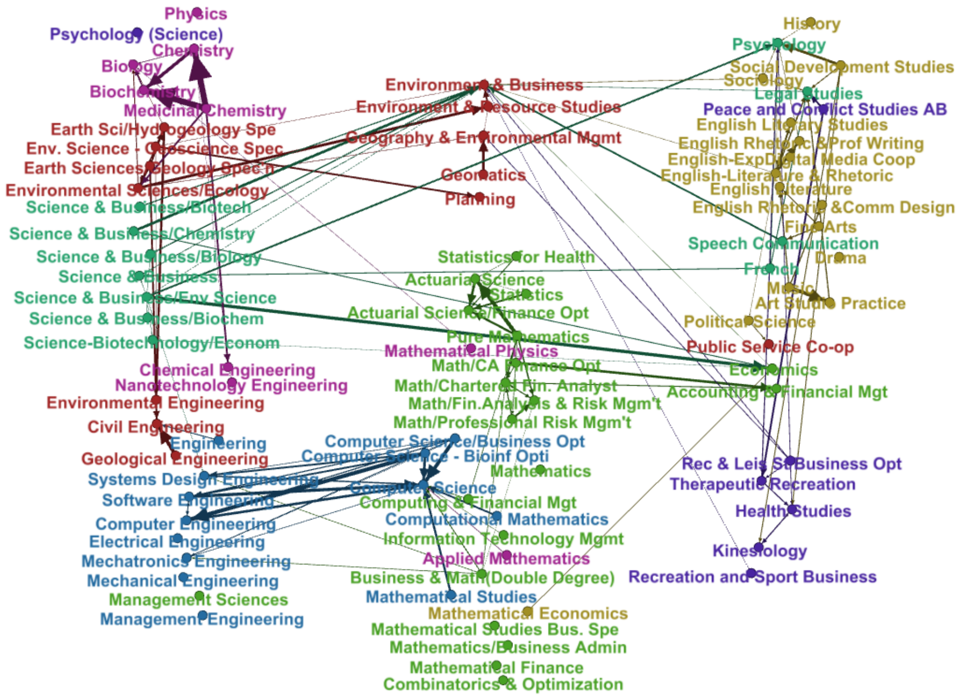


Figure 6: Word cloud of job titles of 85 Civil Engineering jobs that also interviewed students from other programs

interview students from any other program. They are Mathematical Studies/Business, Environmental Science - Geoscience, Information Technology Management, Accounting & Financial Management, Kinesiology, Chemical Engineering, Mechanical Engineering, and Civil Engineering. Upon inspection of the 70 jobs that interviewed only Civil Engineering students, we found that the job titles reflected expertise that is specific to this program, such as “structural”, “field inspector”, “bridge”, “traffic” and “transportation” (see the word cloud in Figure 5). However, the remaining 85 jobs that interviewed Civil Engineering students also interviewed students from other programs, mostly other engineering programs such as Environmental, Mechanical and Geological Engineering. We show a word cloud of these job titles in Figure 6; notice that it includes more general keywords as compared to those in Figure 5. Thus, it appears that there may not be enough specialized jobs for programs such as Civil Engineering and some students within such programs compete for a broader set of jobs.

5. CONCLUSIONS

We presented a data-driven solution towards improving the co-operative education process. We observed that academic programs are typically used by students and employers to advertise and search for jobs, but it is not always clear how one program differs from another, especially given that universities have recently been creating new programs. In response to this problem, we developed a methodology to characterize the relationships among academic programs with respect to the job interviews obtained by students from these programs. The insight behind the methodology was to transform co-op interview data into a *program graph*, which revealed that students from certain programs interview for the same jobs as those from other programs. We proposed graph analyses such as finding communities, finding vertices connected to many communities, and finding vertices strongly connected to their neighbours to describe the program relationships.



Physics
Psychology (Science)
Chemistry
Biology
Biochemistry
Medicinal Chemistry
Earth Sci/Hydrogeology Spe
Env. Science - Geoscience Spec
Earth Sciences/Geology Spec'n
Environmental Sciences/Ecology
Science & Business/Biotech
Science & Business/Chemistry
Science & Business/Biology
Science & Business
Science & Business/Env Science
Science & Business/Biochem
Science-Biotechnology/Econom
Chemical Engineering
Nanotechnology Engineering
Environmental Engineering
Civil Engineering
Engineering
Geological Engineering
Systems Design Engineering
Software Engineering
Computer Engineering
Electrical Engineering
Mechatronics Engineering
Mechanical Engineering
Management Sciences
Management Engineering
Environment & Business
Environment & Resource Studies
Geography & Environmental Mgmt
Geomatics
Planning
Statistics for Health
Actuarial Science
Statistics
Actuarial Science/Finance Opt
Pure Mathematics
Mathematical Physics
Math/CA Finance Opt
Math/Chartered Fin. Analyst
Math/Fin.Analysis & Risk Mgm't
Math/Professional Risk Mgm't
Computer Science/Business Opt
Computer Science - Bioinf Opti
Mathematics
Computer Science
Computing & Financial Mgt
Computational Mathematics
Information Technology Mgmt
Applied Mathematics
Business & Math(Double Degree)
Mathematical Studies
Mathematical Economics
Mathematical Studies Bus. Spe
Mathematics/Business Admin
Mathematical Finance
Combinatorics & Optimization
History
Psychology
Social Development Studies
Sociology
Legal Studies
Peace and Conflict Studies AB
English Literary Studies
English Rhetoric &Prof Writing
English-ExpDigital Media Coop
English-Literature & Rhetoric
English Literature
English Rhetoric &Comm Design
Fine Arts
Speech Communication
Drama
French
Music
Art Studio Practice
Political Science
Public Service Co-op
Economics
Accounting & Financial Mgt
Rec & Leis St Business Opt
Therapeutic Recreation
Health Studies
Kinesiology
Recreation and Sport Business

Figure 4: Clustering of the program graph into seven communities

We applied the proposed methodology on a large co-op data set from a major Canadian university. Our findings and their significance may be summarized as follows.

The clustering and community detection results (Section 4.1) correspond to job categories and academic specializations, which are not always evident from the University's academic structure. This suggests a job classification hierarchy to help advertise jobs to groups of related programs. Our results can also help students plan their academic and employment careers.

In Section 4.2, we identified multi-disciplinary programs which have strong connections to multiple clusters. These results can help students select programs that will give them broad skills and job qualifications, and can help institutions confirm that programs designed to be multi-disciplinary are producing students who qualify (i.e., are able to obtain interviews) for various types of jobs.

In Section 4.3, we identified programs where there were no jobs that only interviewed students from that particular program. That is, students from that program always competed for jobs with students from other programs. The university may wish to attract more employers that offer jobs to these under-represented programs.

6. REFERENCES

[1] R. Barnett. Supercomplexity and the curriculum. *Studies in Higher Education*, 25(3):255–265, 2000.

[2] R. Barnett. Learning for an unknown future. *Higher Education Research & Development*, 31(1):65–77, 2012.

[3] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In Proc. of the International AAAI Conference on Weblogs and Social Media, 2009.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[5] M. Borrego and J. Bernhard. The emergence of engineering education research as an internationally connected field of inquiry. *Journal of Engineering Education*, 100(1):14–47, 2011.

[6] E. El-Khawas. Higher education re-formed: Peter scott (ed.): Falmer press, London, 2000. *Higher Education Policy*, 14(1):93–95, 2001.

[7] Z. Fadeeva, Y. Mochizuki, K. Brundiers, A. Wiek, and C. L. Redman. Real-world learning opportunities in sustainability: from classroom into the real world. *International Journal of Sustainability in Higher Education*, 11(4):308–324, 2010.

[8] A. J. Hesketh. Recruiting an elite? employers' perceptions of graduate education and training. *Journal of Education and Work*, 13(3):245–271, 2000.

[9] Y. Jiang, W. Y. S. Lee, and L. Golab. Analyzing student and employer satisfaction with cooperative education through multiple data sources. *Asia-Pacific Journal of Cooperative Education*, 16(4):225-240, 2015.

[10] D. Kember, A. Ho, and C. Hong. The importance of establishing relevance in motivating student learning. *Active Learning in Higher Ed.*, 9(3):249–263, 2008.

[11] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint 0812.1770*, 2008.

[12] M. E. Newman. Analysis of weighted networks. *Physical Review E*, 70(5):056131, 2004.

[13] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69:026113, Feb 2004.

[14] A. Wiek, L. Withycombe, and C. L. Redman. Key competencies in sustainability: a reference framework for academic program development. *Sustainability Science*, 6(2):203–218, 2011.

[15] A. Wilson. Strategy and management for university development. In *Higher Education Re-Formed*, Falmer Press, pp. 29-44, 2000.

[16] A. Wilson. *Knowledge power: interdisciplinary education for a complex world*. Routledge, 2010.

[17] World Association for Cooperative & Work-integrated Education (WACE). Accessed on 25 Feb 2016, at `www.waceinc.org/global_institutions.html`.