8-2014

Data Analytics of University Student Records

Mark Blaise DeCotes
University of Tennessee - Knoxville, bdecoate@vols.utk.edu

Recommended Citation

DeCotes, Mark Blaise, "Data Analytics of University Student Records. " Master's Thesis, University of Tennessee, 2014.
https://trace.tennessee.edu/utk_gradthes/2806

To the Graduate Council:

I am submitting herewith a thesis written by Mark Blaise DeCotes entitled "Data Analytics of University Student Records." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Jian Huang, Major Professor

We have read this thesis and recommend its acceptance:

Bradley T. Vander Zanden, Wei Gao

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

8-2014

Data Analytics of University Student Records

Mark Blaise DeCotes

University of Tennessee - Knoxville, bdecoate@vols.utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Mark Blaise DeCotes entitled "Data Analytics of University Student Records." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Jian Huang, Major Professor

We have read this thesis and recommend its acceptance:

Bradley T. Vander Zanden, Wei Gao

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Data Analytics of University Student Records

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Mark Blaise DeCotes

August 2014

Acknowledgements

I would like to thank my parents, Mark and Cathy, for always believing in my ability and my committee for their valuable insights. This work is dedicated to my fiancée, Alexandra.

Abstract

Understanding the proper navigation of a college curriculum is a daunting task for students, faculty, and staff. Collegiate courses offer enough intellectual challenge without the unnecessary confusion caused by course scheduling issues. Administrative faculty who execute curriculum changes need both quantitative data and empirical evidence to support their notions about which courses are cornerstone. Students require clear understanding of paths through their courses and majors that give them the optimal chance of success. In this work, we re-envision the analysis of student records from several decades by opening up these datasets to new ways of interactivity. We represent curricula through a graph of interconnected courses, studying correlations between student grades. This opens up possibilities for discovering intellectual prerequisites not shown in the course catalog. Extending this, we define a similarity metric for majors within the university, performing hierarchical clustering to reveal structure within this graph of majors not even present within the catalog. Lastly, we seek to show the temporal development of majors as the network grows through time. Through these approaches, our work provides improvements to current methods of viewing and interacting with student records.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Imagine if you will, the plight of a first year undergraduate with aspirations towards a degree. This student is, in most cases, in possession of a minimal set of skills for tackling the challenging courses they will face. They must learn how to adapt to their new collegiate life and challenging curriculum [4]. If the student is likely to succeed, a proper path through the curriculum should be clearly laid out for them. To this extent, course advising and many other freshmen engagement services are offered. Unfortunately, a student may be advised incorrectly because of either a misunderstanding of the student's own interest or because of faculty or staff misunderstanding of the intricacies within the curriculum [15]. This situation may sound contrived, but it can occur, be it because an advisor overestimates a student's capabilities or because prerequisite courses as laid out in the catalog might not reflect how the curriculum is actualized. Thus, a bevy of issues may remain for students who exhaust all university resource channels. In order to create an environment where students are given the best chance at succeeding, advisory personnel must truly know their institution.

Because of these potential pitfalls in the standard advisory route, resourceful students will often heed the advice of peers in determining an appropriate course schedule. Those who can provide the most compelling advice are often ones with shared personal experience, and inquiring the opinion of an upperclassman can provide information with the highest relevancy. Unfortunately, this solution has two key limitations: *scalability* and *reliability*. The issue of scalability arises because the physical task of interpersonal conversation carries with it a large inherent time cost. Alternatively stated: one simply does not have the time to inquire every student. Additionally, how does one measure the reliability of this information, given that personal biases are commonplace when students reflect upon their particular experiences. Furthermore, one cannot obtain quantitative data via this method. How can we verify the statistical significance of a single student's opinion? With this in mind, imagine that, instead of being limited to pondering the advice of few students, one could base decisions on the cumulative experience of every student who had ever attended the university. Then, imagine that all of these students told the unbiased truth, providing quantitatively verifiable clarity in their responses. The capability to accomplish this lies within a cornucopia of untapped potential: university student records.

This work focuses upon creating a framework for performing meaningful analysis in new ways on a dataset which has been constantly reviewed over decades. Modern data analysis for a multitude of use cases revolves not around fundamental transformations in the statistical techniques for studying data but instead focusing upon how we view and interact with the data. The relational databases used to store information typically accumulate entries over time via numerous transactions which are minuscule in proportion to the sum of their parts. Such transactions carry temporal significance, and typically will form smaller subsets within the database corresponding to certain temporal features such as a transactional timestamp or, for a university database, an academic year. Thus, typical analyses for these records will correspond to small subsets of the data which exhibit some temporal or spatial relationship [12].

University databases which contain historical records of student information are a prime example that which has been analyzed extensively, but only for temporal subsets within the data and mostly for prototypical statistical measurements for administrative progress reporting purposes. These records are an example what is currently popular to call *big data.* In no means are university records immense datasets such as those produced by particle simulations or large web corporations, so they are not *big* in that regard. They instead are *big* in their potential because the intrinsic value they hold is immense. The question then becomes: how can one extract previously unknown features from such a dataset? The answer is to facilitate user interactivity with the data.

Our focus is to elucidate, through interactivity, the true structures of university curricula. In doing so, we bring to light knowledge previously unquantified at best and unknown at worst. We take a fundamentally different approach to our analysis, choosing to view the university as a pipeline which students travel through, meeting certain quality assurance checks at scheduled intervals via their classroom performance. These checkpoints, where measurements are recorded in the form of final grades students receive, can be analyzed. Thus, we view the pipeline as an interconnected network of temporally related checkpoints, determining the relationships between them. It is here that we can infer meaning from the pipeline, such as calculating correlation between measurement distributions at different checkpoints. Empowered by this information, we have the capability to confidently inform not only the resourceful student looking for academic advising but their advisors and administrators as well.

Our answer to the problem is to use statistical analysis of student records to infer correlations between grades in various classes in all majors. By determining which courses are most related based upon student performance, we are able to offer key insights into curricula with no *a priori* knowledge of the specific structure. Through this approach, we can also find relationships between courses which contain very little relatable material. Our system reveals potential course prerequisites which are not listed by the catalog. We also propose a data-driven similarity metric for majors at the university, and use this to develop a hierarchical clustering of all majors. Additionally, we track this clustering through time, from freshmen to senior-level courses, and study how the structure develops. In order to facilitate doing all of this in a useful way, we developed a framework for the visualization of university records to encourage new ways of interactivity with the dataset.

The organization of the rest of the thesis is as follows: Section 1.2 will provide related work in this field. Chapter 2 includes several sections pertaining to the technical background of our work and discusses visualization techniques used. Chapter 3 then talks about the processing and analysis of the data, and Section 3.3 discusses the concept pipeline of our system. In Chapter 4, we present visualization results of our work and initial user feedback. Finally, in Chapter 5, we draw conclusions and discuss directions for future work.

1.2 Related Work

One approach for aiding students stems from the notion that student success can be attributed to overall engagement in courses and activities [14]. The University of Kentucky created a system based on analysis of student information that sought to quantitatively measure student engagement. They coalesced class performance, advising attendance, demographics, campus activity involvement, and other factors into what they coined the "K-Score" [11]. Their approach sought to have students maintain an active role in campus activities by monitoring those who may be struggling academically and intervene. Such an approach may prove hopeful for yielding positive retention results, but they treat the curriculum itself as a black box, instead focusing on its effect over students. As another example, the University of Tennessee recently implemented a system "uTrack" with the goal of successfully guiding students to a timely graduation [19]. The goal is to empower students with knowledge of which courses they need to complete, and when to do so, in order to graduate in a timely fashion. When students are advised, the goal is to help them move down this path leading to success while maintaining their academic interest. The long term success of such programs remains to be seen, but they are strong steps towards maintaining student engagement and encouraging timely graduation.

Another approach taken into providing visual analysis of student data comes from the CourseVis system [17][18]. This tool helped professors who taught distance learning courses gain a greater understanding of how their students interacted and learned through the online system beyond that of their test scores. There are multiple data views offered, including a three-dimensional scatter plot meant to convey information about discussion topic threads for the course. The system also uses what the authors call a Cognitive Matrix for visualizing the correspondence between student performance on quizzes as they relate to the topic covered. It is essentially a heatmap based visualization which organizes students and topics together and encodes student performance through a color mapping. While this work offers visual analysis of student performance, it is limited in its scope, focusing a single courses.

One more example of academic work on the subject involved examining student understanding of a curriculum via exploring comprehension within individual courses. The DynMap tool offered the ability to visually inspect students' understanding and performance within concepts of a course as well as displayed the overall structure of the course topics and their dependencies [20]. This work, similarly to the CourseVis system, analyzed student records and focused upon understanding of individual courses within the curriculum.

While the works presented offer interesting contributions to the topics of curriculum understanding and student retention, they are limited in their scope. The analyses present direct their attention to studying data that mostly arises within individual courses. This work approaches university curricula holistically, studying how performance within individual courses relate to each other and how this information can be used to draw meaning about university majors.

Chapter 2

Background

2.1 Visualization

2.1.1 Graph Visualization

By definition, a graph G can be described as an ordered pair of sets $G = (V, E)$, where V is the set of vertices and E is the set of edges which connect the vertices. Edges are are comprised 2 vertices. If an edge is composed of vertices a and b, then we may represent this edge with the notation (u, v). In an undirected graph, edges have no orientation, thus (v, u) is equivalent to (u, v). One of the most common and intuitive ways to represent a graph is the node-link diagram. Figure 2.1 shows an example graph in this form. In the figure, the vertices are $V = \{u, v, w, x, y\}$ and the edges are $E = \{(u, v), (u, x), (v, w), (v, y), (w, y), (x, y)\}$. This representation mainly stems from how natural it is to preform path tracing via this visualization. However, node-link diagrams can suffer from cluttering and readability issues for graphs with large numbers of vertices or high link density. Ghoniem et al. define link density d in a graph as

$$d = \sqrt{\tfrac{l}{n^2}}$$

where l is the number of links and n is the number of vertices (or nodes) in the graph [7]. For all graphs, d will take on a value between 0 and 1. As d approaches 1, the readability of the node-link diagram rapidly declines. In order to provide an

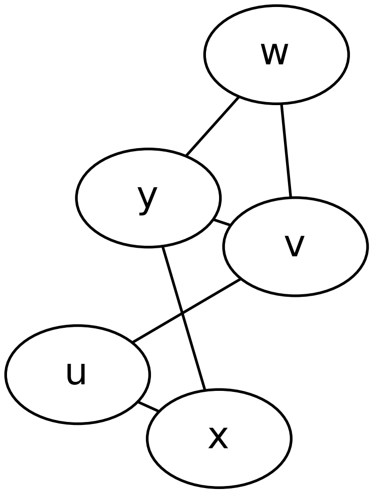


Figure 2.1: An undirected graph visualized via node-link diagram.

informative visualization, the issue of readability based on node density must be dealt with.

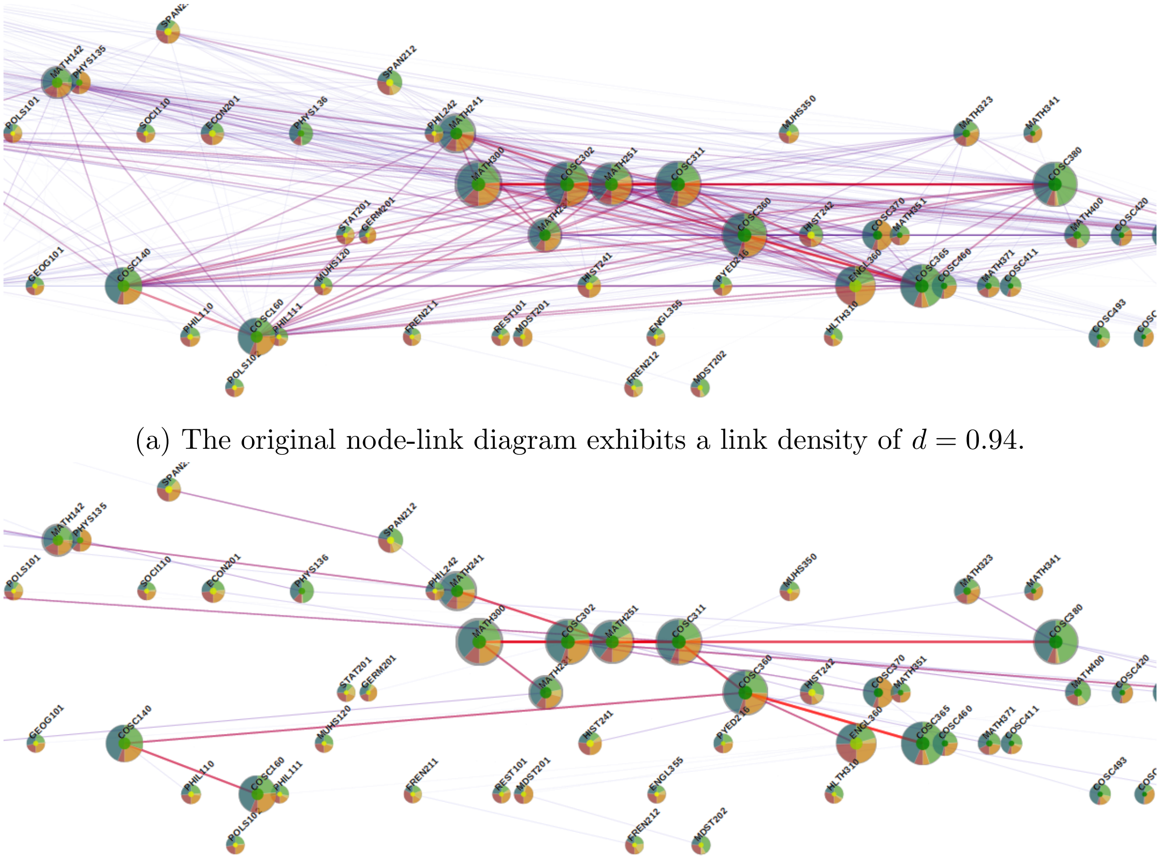
Providing visual clarity within the clutter of highly connected graphs is one topic out of many within the well researched field of graph visualization [8]. As a graph becomes fully connected, the number of edges approaches $O(|V|^2)$, which corresponds to a link density value of $d = 1$. For such graphs, naively displaying every edge within a node-link visualization quickly leads to substantial visual clutter. Attempts at rendering as such show little information other than the high connectivity of the graph. An example of this for our work can be seen in figure 2.2a.

Many node-link layout models are available, including the popular choice of force directed[6]. Experiments with utilizing the D3.js [3] library implementation of a force directed layout aided in extracting certain features from the graph, but still did not solve the issue of high link density and visual clutter. Edge bundling [9] is another technique for reducing visual clutter from large numbers of edges. The idea is to bundle adjacent edges together into a single link by combining them using cubic B-splines where all corresponding vertices are used as control points. Additionally, this has the effect of creating an implicit edge hierarchy, where the generated splines are clusters of edges. This technique is valuable, but we ultimately decided on a different approach because of our desire to visualize node connectivity in a pairwise fashion. The solution that we implemented utilized computation of the spanning tree, keeping only the links with the highest intrinsic value for each node pair. A spanning tree of a graph is a non-unique subgraph which contains enough edges to link together all vertices. The minimum spanning tree of a weighted graph (a graph with values associated with edges) is a spanning tree with the smallest possible summed edge weight. It is an important tool in many areas, including computer communication networks and wiring connections [5]. Many algorithms exist for computing the minimum spanning tree of a graph. We used Prim's algorithm for computing a spanning tree such that the summation of link weights was maximized. The results of this can be seen in Figure 2.2b.

Another way to visualize a graph is through an adjacency matrix representation. In an adjacency matrix, the cell in row i and column j represents an edge connecting vertices i and j. One advantage that a matrix-based approach has over a node-link diagram is that permuting the rows or columns of the matrix is a straightforward task that can yield impressive results for revealing structure within the graph. It is also possible to argue that topology of a graph, such as clusters or groups of vertices with a strong connection, may be more readily understood from an adjacency matrix [7]. An example adjacency matrix may be seen in Figure 4.2.

2.1.2 Parallel Sets Visualization

Visualization techniques generally applied to continuous datasets typically do not bode well when extended for categorical data. Typically, issues arise because of the discrete nature of categorical data, with visual metaphors falling short because of



(a) The original node-link diagram exhibits a link density of $d = 0.94$.

(b) Reducing the number of visible links lowers the link density to $d = 0.11$.

Figure 2.2: Mitigating visual clutter within a node-link diagram by only showing links of highest value.

their reliance on continuous domains such as space or color. Bendix et al. created a visualization technique called parallel sets with the goal of properly encoding categorical data with appropriate visual metaphors [1]. It is based upon the parallel coordinates technique, which represents the N-dimensional data tuple C with data values $(c_1, c_2, ..., c_N)$ by points on N parallel axes. The points are then connected to form a polyline [21]. Parallel sets extends this functionality to categorical data by replacing the continuous axes with sets of color coordinated regions that represent the categories. Thus, a visual metaphor for the discrete nature of the dataset may be more easily established by choosing an appropriate discretized color set. For our work, the discrete sets are various grades received by students, while the axes represent the

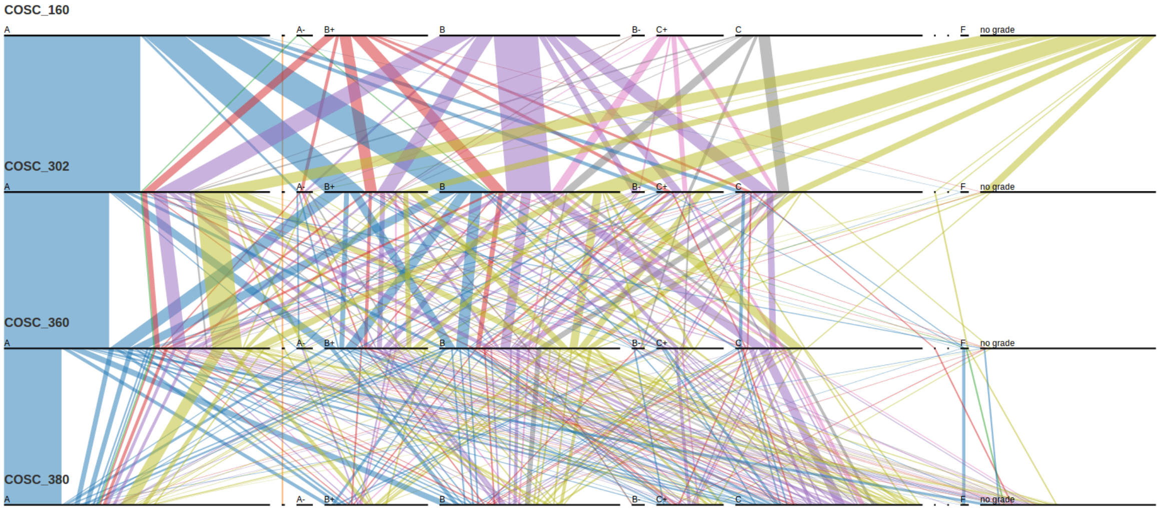


Figure 2.3: Parallel sets visualization for student grades from four Computer Science courses.

course in which students received a particular grade. Figure 2.3 shows an example of our implementation. Sets are colored according to the grade received in the first course (*COSC_160*). General trends are visible such as a lower average for *COSC_380* or a lower population of students who took *COSC_160*.

2.1.3 Radial Tree Visualization

Hierarchical data is ubiquitous in society: from the directory structure within a computer file system to corporate structuring within organizations to software library dependencies. The natural analogue to this hierarchical structure is the tree. This analogy is oft used in computer science because the concepts of roots, branches, and trunks are usually first learned via studying real-world trees. Traditionally, when visualizing hierarchical data as a tree, the de facto standard is to represent the data via a directed graph beginning with a fixed root node and continuing with child nodes who are connected to parent nodes via lines (see figure 2.4). This node link representation is used because it is natural for people to trace paths through the hierarchy. The goal of any hierarchical tree layout is to provide visual analogies that effectively translate the structure of the tree [8]. A radial tree is a visualization

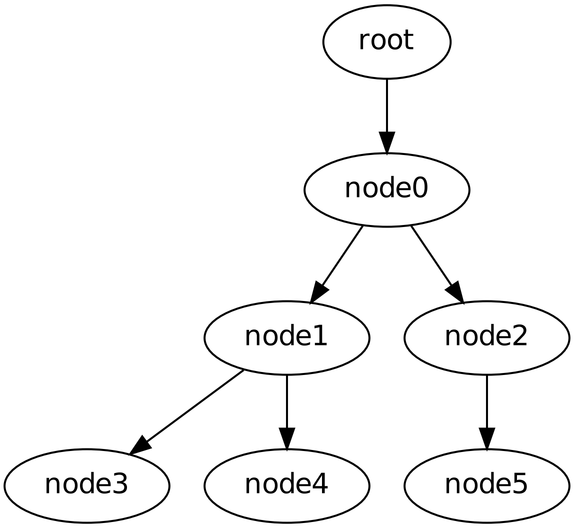


Figure 2.4: Hierarchical data represented as a tree, visualized as a directed graph.

technique which uses concentric circles to convey tree depth. For a length l, the radius of the concentric circle which a node at depth d will be placed is calculated as $l * d$. The root node, with depth 0, will be placed in the center, with child nodes expanding outward in the layout. The reasoning for using this layout is to provide a structure which scales well as the number of nodes increases. Our work utilizes a radial tree for visualizing the hierarchical structure of majors at The University of Tennessee, according to a similarity metric which we developed. This metric will be discussed in detail in section 3.1.3.

2.2 Student Records Dataset

Student records are stored by universities in databases. Because of the large size and complexities of many universities, data records can be scattered in databases under the supervision of many departments. Records of acceptance into the university may be kept separate from information about student grades or their declared major, as permissions for the information might lie under different departments. The end result is a system which proves difficult for use other than typical storage and retrieval purposes. For our work, we were able to gain access to multiple sources of student information from many different university databases. Table 2.1 shows information about the raw data. With the data being focused on individual students, our first goal was to cluster students together based on a common trait. Students have many pieces of information which may relate them, including major of interest when accepted into the university, major a student eventually graduated with, standardized test scores, college credits when accepted, *et cetera*. We decided to focus on individual majors within the university and thus when partitioning the student body, we did so by graduating major. This limits our population to those students who received a degree from the university. For future work, including information from students who did not eventually graduate from the university may help to understand the influence of students who withdraw.

Table 2.1: Information about raw data used. The data for our work was from the past 16 years, containing information from 144798 students and covering over 350 majors.

Description	Number of Entries	Size (MB)
Graduation Records	100239	33
Courses Taken	4485377	524
Grades Received	4723835	461
Admissions Records	399989	119
Major Code Description	2537	.2

Once a student population has been extracted, it is then necessary to extract relevant information about their course performance. This involved determining all courses taken by the population. Being aware that curricula can and should change over time to adapt to new and evolving fields of study, a temporal parameter was included to extract only information from a given range of time. We can apply a filter to the data to only extract information relevant to the desired range. Because of general education requirements, there are many courses taken by only a small number of students. For course to course comparisons, we filter out courses taken by less than 15% of students in order to retain information about some of the lesser taken courses but to exclude the vast majority of this noisy portion of the data. Having determined which courses to include for each major, the remaining portion of data extraction involves associating students with their grades for courses taken, as well as recording the semester when it was taken to give temporal context. Information about when courses are taken is critical for creating an intuitive visual representation of course progression paths later on.

Chapter 3

Data Processing

3.1 Statistical Analysis

3.1.1 Correlation Coefficient

For analyzing collections of grades for two courses, we utilized the Pearson's correlation coefficient (PCC), which measures the linear dependence between two variables. This has the effect of quantifying how grades in one course vary in correspondence to grades in another course. This measure is advantageous over regression analysis because the grade samples should be viewed as varying independently. Let X be a collection of grades for course A, and Y be the collection of grades for course B. For these sample populations, the PCC $r_{A,B}$ can be described as the sample covariance of X and Y divided by the sample variance of X multiplied by the sample variance Y. Thus we have,

$$r_{A,B} = \frac{\sum_{i=1}^{N_{A,B}} (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N_{A,B}} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{N_{A,B}} (Y_i - \bar{Y})^2}} \tag{3.1}$$

where $\bar{X}$ and $\bar{Y}$ are the sample means for X and Y, respectively, $N_{A,B}$ is the number of students who took both course A and course B, and each X_i, Y_i are specific grades.

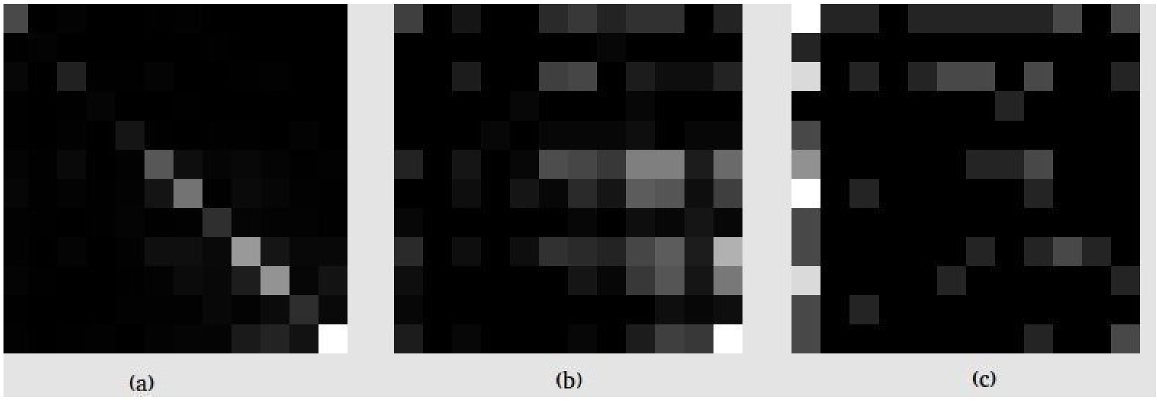


Figure 3.1: Histograms of course grades for three course pairs. Each box represents a specific grade pair for the two courses. Grades from F or W (course withdrawal) to A are arranged from top left to bottom right, respectively. Whiter color represents a higher count of students.

3.1.2 *C-Value*

By using the PCC, we can see relationships between courses based upon how students performed within both of these courses. In order to avoid the possibility for certain courses to have a higher correlation coefficient because of a smaller sample size for a particular course pair, we perform a scaling on the PCC for our purposes. We call this resulting term the C-value. So for each pair of courses A and B, we have,

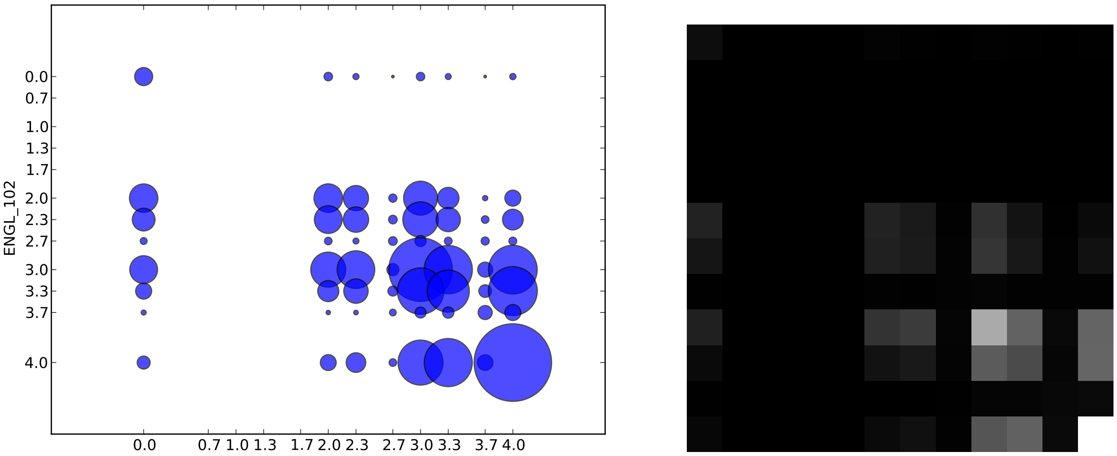
$$C_{A,B} = N_{A,B} \cdot r_{A,B} \tag{3.2}$$

Figure 3.1 shows histogram layouts for counts of various grades received in three course combinations and shows potential linear relationships between grades received in various course pairs, which are quantified by the C-value. This is displayed as a heatmap, which is a visualization technique similar to the adjacency matrix, that succinctly reveals potential structure in a data matrix [23]. The top left corresponds to grades of F or W (course withdrawal) in both courses, and the bottom right corresponds to grades of A. White indicates a larger number of students, while black indicates no students. Figure 3.1(a) shows Spanish 111 and Spanish 112; these courses display a nearly linear correlation with the grades students receive. Figure 3.1(b) shows Spanish 111 and Chemistry 120; less correlation is shown in the grades for these two courses, but there is still a relationship between high grades in both. Figure 3.1(c) shows Spanish 111 and Spanish 150; here we see that students who took both of these courses tended to perform poorly, and that the majority of students failed or withdrew from Spanish 150 no matter their grade in Spanish 111. Utilizing visualization such as heatmaps provides an indication of the full structure of a data matrix to aid in the analysis of the PCC. One can contrast a heatmap with a scatterplot for visualizing the same data (see 3.2). The scatter plot shows the distribution of the grade pairs by encoding it into the radii of the plotted circles. This is an advantage over the heatmap, though the scatter plots are less frequently used because it can become cluttered.

The C-value gives us a numerical measurement of the similarity between two courses. In our work, we compute the C-values for every course combination within every major. Thus, we create a graph where the vertices are courses and the edges are the C-values between these courses. This representation allows us to leverage the vast research in graph visualization for aiding in the understanding of the university curricula.

3.1.3 *M-Value*

With the C-value, we obtained a course to course similarity metric for the curricula of individual majors at the university. This led to the question of whether or not we could formulate an analogous major to major similarity metric. This problem required a different approach, however, because student grades for individual courses no longer held as much value when working on this macroscopic level. To call two majors ”similar” we needed to determine what exactly defined a major. In order to keep results focused around data-driven methods, we decided: *a major should be defined by the courses taken by students who achieve a degree in said major.* This straightforward definition follows the notion that in order for a student to qualify to



(a) Here circle size represents the number of students who received a particular grade pair.

(b) Here color value represents the number of students who received a particular grade pair.

Figure 3.2: Visualizing grades received in English 101 and English 102 as a scatter plot and a greyscale heatmap.

graduate with a particular degree, they must fulfil all criterion for their curriculum with satisfactory results. It also allows for data to reveal unexpected results, where a significant number of students may have taken particular courses which fulfilled no requirements towards their graduation.

Thus, for two majors, we define the similarity value between two majors as the likelihood that a student from each major takes the same course. One can envision this concept as asking the question: how many shared courses will students from two majors have? More formally, for majors X, Y, we can define the similarity measure $M_{X\to Y}$ as :

$$M_{X\to Y} = \sum_{c_i \in Classes} \frac{s_{Y_i}}{|S_{c_i}|_2 |Y|} \tag{3.3}$$

where $c_i = \{s : s\ is\ a\ student\ in\ course\ i\}$ is the set of all students in course i, s_{Y_i} is the number of students in c_i from major Y,

$$S_{c_i} = [s_{m_1}, s_{m_2}, ..., s_{m_n}]$$

is a vector of counts of students from each major in c_i, $|Y|$ is the total number of students in major Y over all courses, and

$$|S_{c_i}|_2 = \sqrt{\sum_{k=1}^{n} s_{m_k}^2} \tag{3.4}$$

is the l^2 or Euclidean norm of the vector S_{c_i}. Now, this one-way M-value calculation is not guaranteed to be symmetric, thus to generate a symmetric distance measure, we perform a mean calculation:

$$M_{X,Y} = \frac{M_{X \to Y} + M_{Y \to X}}{2} \tag{3.5}$$

Note that our calculation of the M-value for two majors depends only upon the subset of courses which we use from the curriculum. Because of this, we can partition courses by which year of study they fall into: freshman, sophomore, *et cetera*. It is then possible to see the similarity of majors evolve as students progress deeper into the core curricula.

3.2 Hierarchical Clustering of Majors

We can view the M-value as a distance metric which can holds the edge weights for a fully connected graph where university majors are the vertices. Doing so, we can represent this graph with a similarity or distance matrix. Thus, we can view the problem of determining similar majors as a clustering problem, and build a radial tree from the clustering results. Building upon knowledge gained from the C-value analysis, we can then analyze not only what courses a student is likely to succeed in but what majors as well.

Our clustering approach uses an agglomerative strategy which groups majors according to their M-value. This has the effect of building up a tree of clusters, and provides various levels of granularity with which to perform an analysis [2],[22].

Algorithm 1 An agglomerative, hierarchical clustering algorithm.

```
function CLUSTER(DistMatrix)
    Initialize all elements as singleton clusters
    while there is more than one cluster do
        Find largest pairwise $M$-value between clusters
        Merge the two clusters and average their link values
    end while
end function
```

The algorithm for agglomerative hierarchical average-linkage clustering is shown in Algorithm 1. Visualizing the results of our M-value clustering can be seen in Figures 4.6a and 4.7a.

As mentioned in Section 3.1.3, our calculation of the M-value depends solely upon the subset of university courses used as input. Because of this, there is an implicit temporal attribute associated with the distance matrix used as input for clustering. Furthermore, as an alternate clustering approach, we could plant seeds of our tree by clustering M-values for freshmen-level courses and let the tree grow and develop through further clustering using courses that are sophomore, junior, senior-level and beyond. In this sense, we can view majors at the university in a more dynamic way, studying their temporal features instead of defining them based explicitly upon the curriculum as laid out in the catalog. All students may begin upon the same path when they enter the university, but with each passing semester, their studies become more unique. Thus with hierarchical clustering combined with our M-values, we are able to extract structure from an unstructured set such as the various majors at a university.

3.3 Parallel Data Processing

For the course to course similarity metric, our problem involves calculating the C-value for every combination of courses extracted during the first part of the workflow. These operations can be performed independently, thus lending some freedom in the

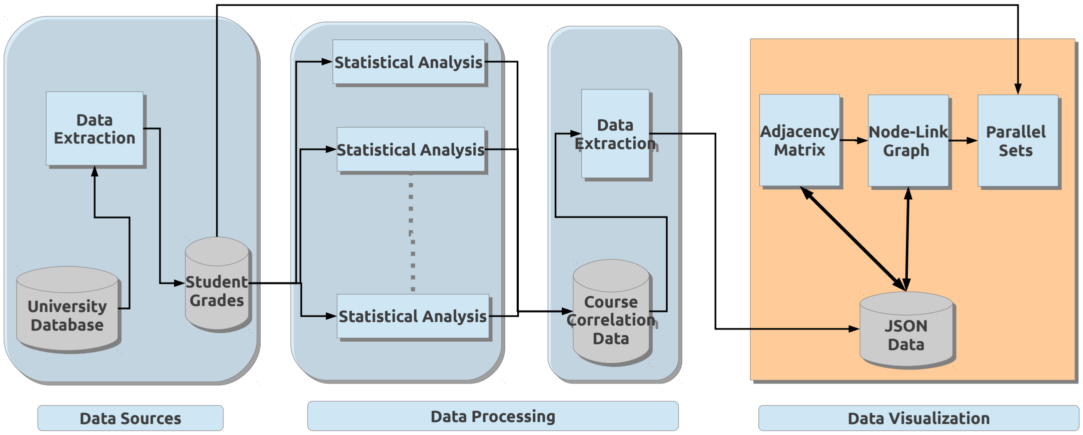


Figure 3.3: Concept pipeline of our system.

way processing may be accomplished. This led to the development of a simple parallel processing system for performing the calculations. Looking at Figure 3.3, we see that many instances of our statistical analysis code may all run simultaneously. Each of these processes extracts the necessary data they need, performs any data validation necessary, calculates the C-value, and emits output to a designated file. These files correspond to specific courses, with each line in the file containing information about correlation and the number of students who took this particular course pair. This parallelism utilizes the file system for the temporary storage of information before the results are pushed down the pipeline. Results are then coalesced and passed on for interactivity within the browser.

Because of our design, we are able to perform the statistical analysis of our workflow in negligible time. When the number of courses is average size, say 100, we can calculate all possible C-values in seconds (See Table 3.1 for more information). Note that the number of correlations computed is $O(n^2)$ where n is the number of courses. This processing time stems almost exclusively from the initial data extraction. A possibility for further improvements in the runtime of this system would be to partition the original raw data by date, thus allowing queries seeking information pertaining to more recent records to be completed quicker. We show improvements seen with this approach in Table 3.1. If real time interactivity with the data were required, it would be necessary to import the data into a relational database structure to allow for smaller, quicker queries.

Table 3.1: Timings (in seconds) for our framework on a machine with 12 x Intel Xeon E5645 with clockspeeds of 2.40GHz.

Version	Min	Max	Mean	Median	σ
Original Data	11.805	49.062	13.514	12.387	3.378
Reduced Dataset	4.003	22.787	11.760	10.958	2.256

Chapter 4

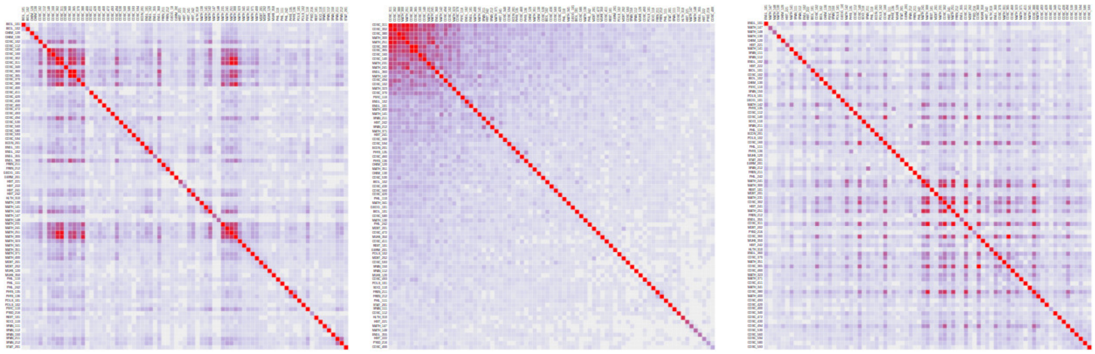
Data Presentation

4.1 Course to Course

4.1.1 Adjacency Matrix

Initial visualization work focused on matrix-based methods of analyzing the course to course correlation graphs. One advantage of an adjacency matrix view is that groupings within the data become clearly visible with certain sorting patterns. Also, the entirety of the graph may be succinctly viewed when laid out as a matrix. To more easily convey meaning in the adjacency matrix, coloring and opacity of the cells were modified to better reflect the C-values used as edge weights. The cells were colored from blue to red and from transparent to opaque, representing low to high C-values, respectively Additionally, we chose three parameters with which to sort the rows and columns in the adjacency matrix: alphabetical based on course abbreviation, C-value summation, and temporal ordering based on consensus from the data. Our web-based adjacency matrix visualization is generated using JavaScript and D3.js [3]. All three of these views can be seen for the Computer Science major in Figure 4.1.

Interestingly, the most intuitive sorting method for this data, alphabetical ordering, turned out to be a great clustering method as well. Courses under the same base abbreviation tended to show higher relationships between their grades



(a) Alphabetical (b) Sum of C-values (c) Temporal Order

Figure 4.1: Adjacency matrix view of the Computer Science course curriculum.

than with others. In Figure 4.1(a), the top left of the matrix shows a cluster of highly correlating computer science courses. This means that for computer science students, the grades they received for some courses in their major correlate with how these students perform in other computer science classes. There is also one more main grouping of courses with high correlation located lower in the matrix. These courses cover mathematics topics such as linear algebra, introduction to abstract mathematics, multivariate calculus, and differential equations. By inspection of the matrix, we can see that these mathematics courses also have a strong correlation with grades students receive in computer science courses.

Sorting the courses based upon their C-value summation reveals courses within the curriculum whose grades have the highest correlation with all other courses. Ordering shows largest to smallest, from left to right and top to bottom (see Figure 4.1(b). This reveals which courses' grades have the highest total correlation with all other classes. The courses whose grades give the highest predictive insight into how students will do overall within the rest of the major are shown in order from greatest to least. Even considering that the PCC is not a general-purpose predictive metric, in this work it offers new insight and is able to quantify the relationships between student grades in courses.

The final sorting method offered is a value assigned to each course based on when on average a student will take it. Organizing the adjacency matrix in this way reveals

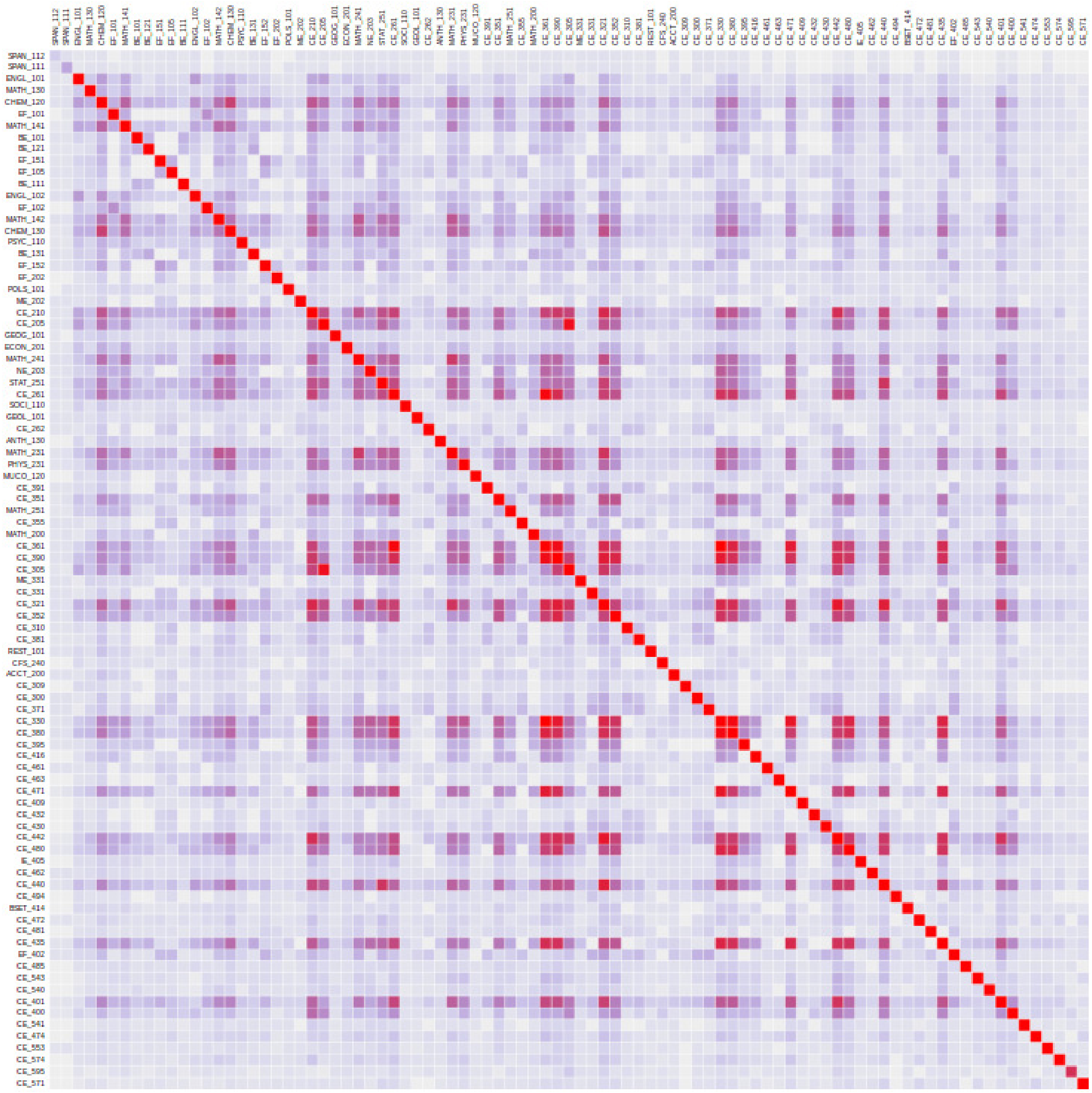
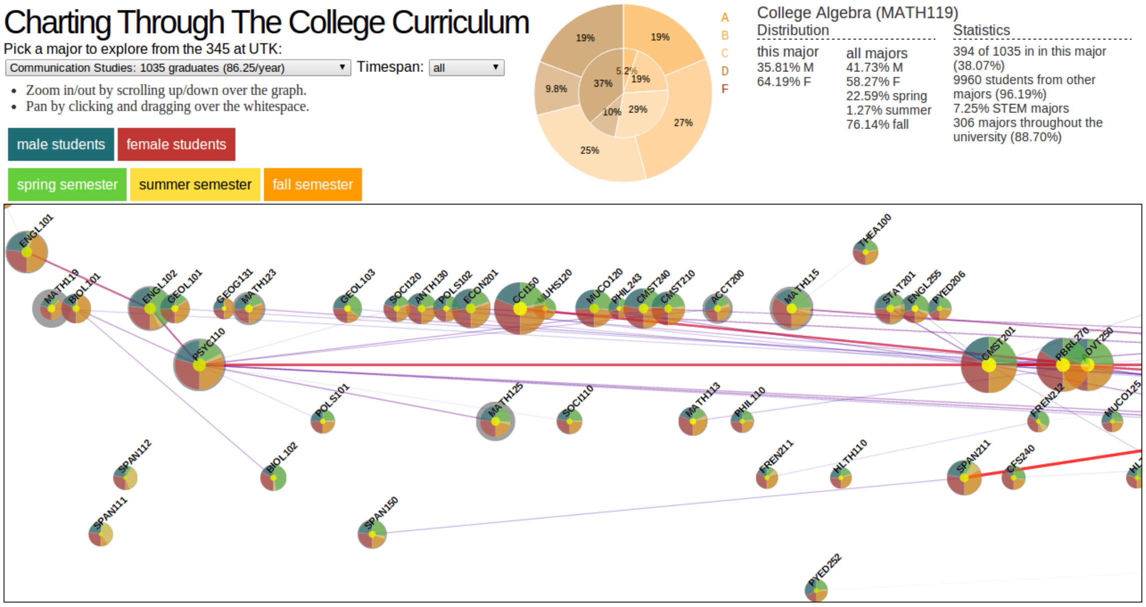


Figure 4.2: Temporal sorting view of courses for the Civil Engineering major.



Charting Through The College Curriculum

Pick a major to explore from the 345 at UTK:

Communication Studies: 1035 graduates (86.25/year) ▾ Timespan: all ▾

- Zoom in/out by scrolling up/down over the graph.
- Pan by clicking and dragging over the whitespace.

male students | female students

spring semester | summer semester | fall semester

College Algebra (MATH119)

Distribution

this major	all majors
35.81% M	41.73% M
64.19% F	58.27% F
	22.59% spring
	1.27% summer
	76.14% fall

Statistics

394 of 1035 in in this major (38.07%)
9960 students from other majors (96.19%)
7.25% STEM majors
306 majors throughout the university (88.70%)

Figure 4.3: Exploring the Communication Studies curriculum.

the temporal context of the data. Figure 4.1(c) shows this view for the computer science major. One can think of time passing from left to right, top to bottom within this view. We can see that the majority of higher course correlations occur within the range of the graph from about halfway to three fourths down. This lets us gauge that the curriculum begins to show strong relationships in course grades around the sophomore to junior level. Some courses taken earlier on, such as the freshmen and sophomore level computer science classes, reveal themselves as darker lines within the upper portion of the matrix. This differs from the civil engineering major, shown in Fig. 4.2, which reveals a more even distribution of highly correlating courses over the lifetime of student's studies. The trend shown for civil engineering courses also shows a larger number of correlating classes than computer science. Revelations into the underlying structure of a major offered by our work allow for analysis by department administration to see if students progression through the curriculum matches the plan set forth when designing it.

4.1.2 Node-Link and Grade Distribution Chart

In addition to the adjacency matrix view, we visualize the C-value graph as a node link diagram. Within this view, edge weights (which are the calculated C-values) once again are encoded using coloring and opacity. Link thickness encodes this value as well. Node diameters are determined by the number of students who took a particular course. This allows visual comparison of differences in average student throughput for individual courses. Within each node, there are three concentric circles. The outermost grey border around nodes encodes the number of students who failed or withdrew from that course. On left side of Figure 4.3, we can see the ring around the course Math 119 (College Algebra) for the Communication Studies major. This denotes that a large percentage of students who eventually graduated with a degree in this major failed or withdrew from Math 119. The middle portion of each node is partitioned to highlight both gender demographics and semester distributions for the course. Courses which are offered mainly in the fall or spring semesters may be viewed as bottlenecks for the curriculum pipeline. It is usually important for the courses which have the largest C-value summations to be offered both during fall and spring semesters to prevent students from falling behind. Finally, the third and innermost layer of the node encodes the "STEMness" of the course. This is a measurement of what percentage of students who take this course go on to earn a degree in a STEM-designated degree program [10]. A course can have 0% to 100% "STEMness", which corresponds to coloring from yellow to green, respectively. Contrasting Figures 4.3 and 4.4, we can see that in 4.3, the majority of courses are yellow, whereas in 4.4 they are green. Thus, we can conclude that Computer Engineering students take more STEM courses than Communication Studies students.

Meaningfully displaying edges is a difficult aspect of node-link diagrams, but it does not encompass the entire problem [7]. Node layout is very important as well, so we looked to the data to help decide an informative representation. The temporal features of the university courses allowed us to fix the horizontal positioning of all nodes in a linear, timewise manner. Thus, every node has a unique horizontal location based upon its temporal attribute. For any node, all courses to its left are generally taken before it and all courses to its right are taken after it. This is based upon a consensus derived from the dataset which quantifies the "average" semester when a course is taken. For vertical positioning, we create a central focus for the top k courses, which is a user-adjustable value. By default, we choose the top $k = 6$ courses based upon C-value summation. This central line forms the skeleton of the graph. All other nodes are placed vertically according to their degree of separation from the skeleton courses within the minimum spanning tree. This allows for course progression paths to be revealed and potentially hidden dependencies to surface. One such example of hidden connections being discovered within a curriculum was the strong correlation for Microbiology students between two courses: Biochemistry I and General Genetics. These two courses have topics which at the surface seemed different, yet grades showed a strong correlation in student performance in these two courses. Discussion with microbiology students revealed that although the two topics do differ in subject matter, they both require similar thought processes, placing emphasis on critical thinking and understanding over memorization. Our approach allows the data to determine relationships between courses beyond that of what the catalog lists or what preconceived notions one might have.

Further exploration of a particular course may be accomplished through the user hovering their mouse over a particular node. On a mouseover event, the top region of the layout displays grade distributions, gender ratios, "STEMness", the number of students from the current major who took this course, and the total number of majors students who took this course graduated in. Looking at Figure 4.3, we see grade distributions for Math 119 taken by Communication Studies majors. The central focus of the top display is a two-layered pie chart. The inner layer visualizes the grade distribution for students who took this course and graduated in the current major being displayed. The outer layer contrasts this with the grade distribution for students of all majors who took this course. For Communications Studies students,

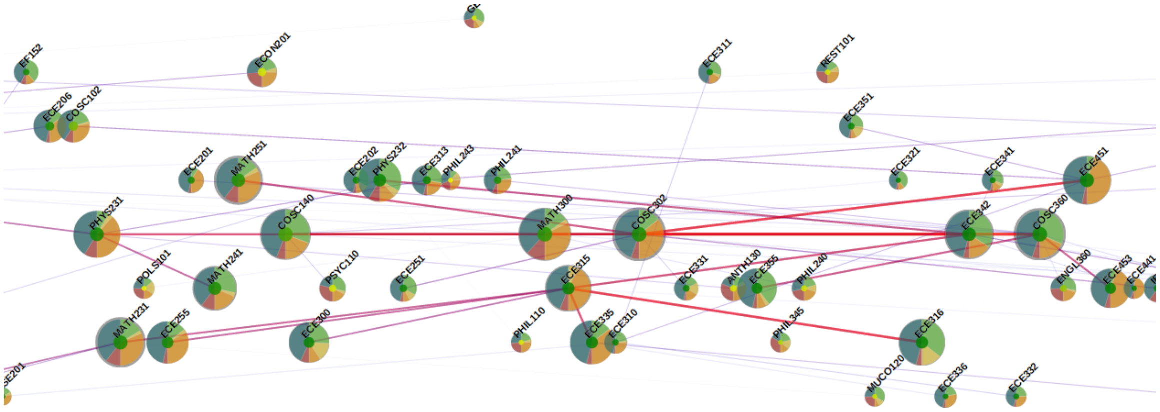


Figure 4.4: The "core" of the Computer Engineering Major.

we can see that nearly 50% receive either an F, D, or withdraw. This is a substantial finding. From this, we discovered that Communication Studies is a backup major for students who wish to pursue a degree in Advertising, and Math 119 is used as a measuring bar by academic advisors to determine if it's in a student's best interest to attempt to pursue an Advertising degree. Our tool was able to showcase this anomaly by providing an overview via the node-link diagram and allowed for deeper analysis by drilling down into the grade distributions.

4.1.3 Initial User Feedback

To test the usefulness and accuracy of our work, we asked students to recommend classes from their major. Then we compared their recommendations with those displayed in our visualizations. Because of the authors' first-hand knowledge of the computer science curriculum at the university, drawing meaning from these diagrams was very intuitive. It was easy to locate computer science courses that teach data structures, algorithms, and systems programming because the visualizations matched up well with our experience within the program.

The Computer Science curriculum at the University of Tennessee is a complex. Not only has the curriculum changed three times within the last ten years, but there is an option to dual major in Mathematics, among others. The most recent change requires

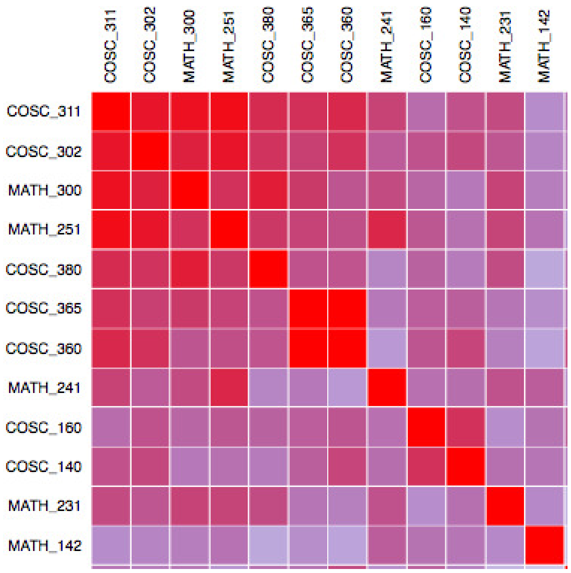


Figure 4.5: Computer Science classes, sorted based on the sum of all C-values

students to pick from almost 140 courses that span 29 different departments. As if that was not enough, many classes offer alternative versions in other departments to allow students a more personalized path of study. All of these options make it difficult for faculty advisors to guide students into the courses that are best for them. Students rely mostly on the advice of their peers who have first-hand experiences with courses.

We sought opinions about classes from computer science upperclassmen. Students with a strong math background found the class COSC 311, which covers discrete structures and includes an introduction to combinatorics, very elementary. Students with a background strictly in programming found the class extremely challenging. As shown in Figure 4.5, student performance in COSC 311 has a very high correlation with their performance in mathematics classes. Seeing the high value that mathematics courses have within the curriculum came as a pleasant reinforcement to the notion that the Computer Science program is composed of more than just programming. Theoretical Computer Science courses were among the highest correlating courses as well. The results from our work differed from the information we gathered from sample computer science students, who believed COSC 311 was among the top three most useless courses in the curriculum. This is a jarring example of the discrepancy that can exist between data and hearsay. While the records indicate students who do well in combinatorics or abstract mathematics will generally do well in other computer science courses, the consensus from students is that these courses offer little to the overall education of the student.

4.2 Major to Major

4.2.1 Radial Tree

Using the results from hierarchically clustering majors based on their M-values and the derived structure, we can visualize the network of majors to see what groupings they form. By basing this structure on a data-driven metric, we can view the university majors and their relationships with each other in an unbiased way. Note that in Equation 3.3, the similarity metric for two majors depends upon the number of shared courses for students from those majors. And if two majors have a high M-value, they are more likely to be paired near each other. This naturally ties to a tree analogy, where we can expect to see branches of groups of majors forming. One can liken this to a dendrogram of species, where similarities might be determined by rRNA sequences, which places similar species near each other. So, to leverage the power of a dendrogram while still attempting to maintain as compact a layout as possible, we choose to visualize these groupings using a radial tree.

4.2.2 Standard Clustering

Looking at Figure 4.6a, we can see the entirety of majors and concentrations offered. Coloring from yellow to red indicates a low to high Computer Science (CS) M-value, respectively. In the middle right portion, we can see a number of majors related to CS denoted by their orange coloring. Figure 4.6b shows a zoomed image of this region. Electrical and Computer Engineering, along with Mathematics, form the clusters in this region. Upon further inspection, however, we can see that Marketing and Statistics group near Mathematics and CS as well. Thus, CS is related to Statistics through a common connection with Mathematics. Such relationships between majors are not available or discernible from the catalog.

Within the lower right region, there are other majors colored yellow-orange, denoting a similarity to CS. This branch, shown in Figure 4.6c, contains Honors Computer Science, as well as Astronomy, Geology, Environmental Studies, and various Physics concentrations. Through our hierarchical method, we see that Honors CS actually clusters more with these other majors than to non-honors CS. This contrasts with Honors Computer and Electrical Engineering, which both clustered closely to their non honors associated degrees. Additionally, Engineering Physics Honors groups together with CS, while the non-honors major groups with Astronomy and to a lesser extent with Honors CS. From this, we can see how subtle changes, such as taking honors courses to fulfill degree requirements, can substantially alter the majors which students will encounter in their courses.

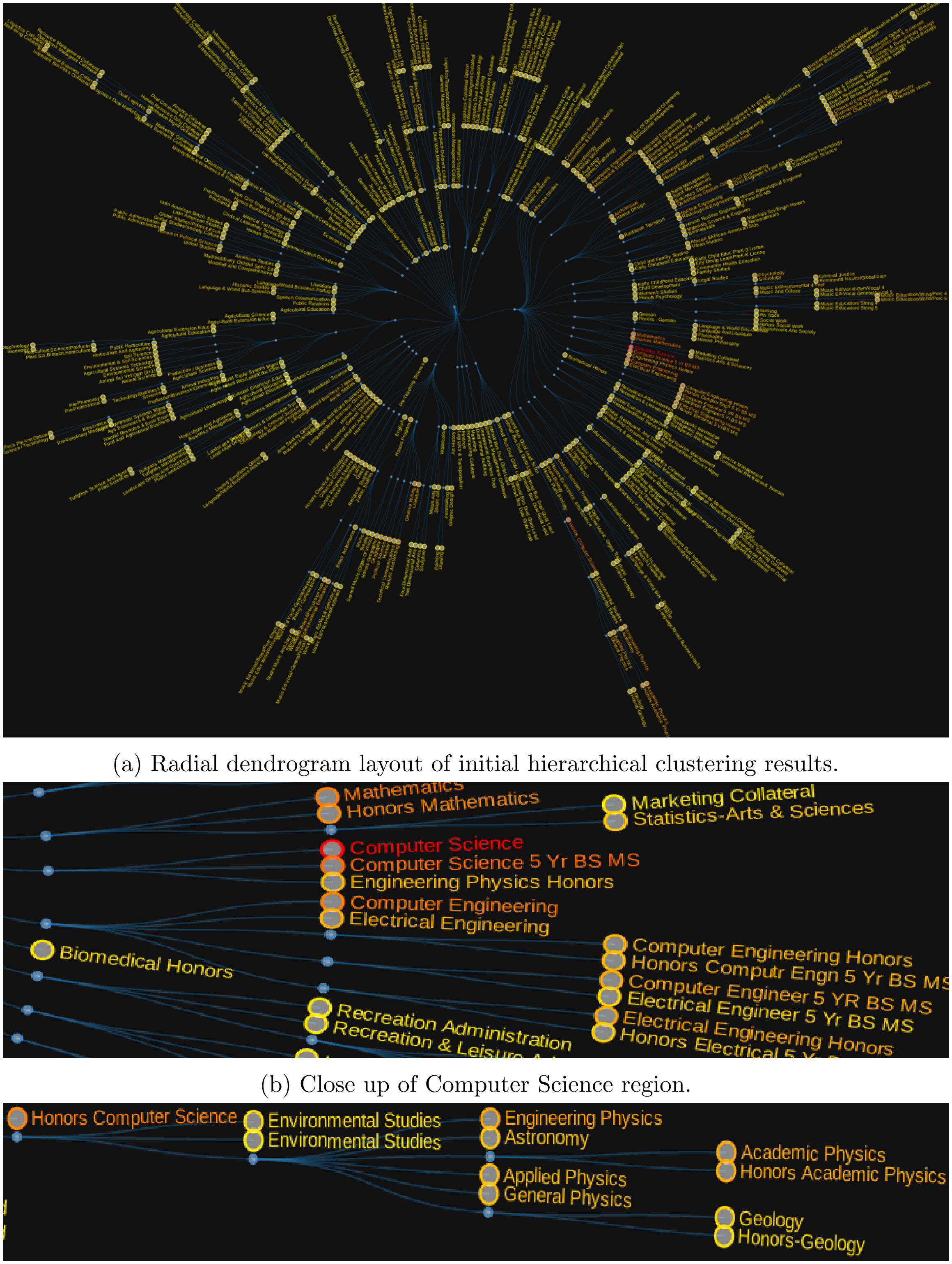
The results from clustering majors based on their M-values provide us with both expected and unexpected groupings of majors. There are straightforward branches of business concentrations, where Logistics, Information Management, International Business, *et cetera* group together. There are also pairings such as Civil Engineering and Construction Science, Advertising and Public Relations, or Horticulture and Soil Science that might not come as a surprise, but, within the larger picture, these branches coincide with surprising majors such as Sport Management, Africana Studies, and Animal Science, respectively. One can view branches within the tree as collections of majors which might be common to switch between based on courses taken by students within such majors or as common dual degree paths. From the tree, we can infer that CS students tend to take some Music Education courses as well, stemming from a common minor chosen by these students.

4.2.3 Path Projection Clustering

As mentioned in Section 3.2, because the hierarchical clustering depends solely upon which courses are given as input, we can derive temporal meaning through partitioning courses according to their temporal attributes. For the results discussed thus far, clustering was performed without respect to whether courses tended to be first year, second year, or beyond. If we wish to look at clustering majors with a greater emphasis on the temporal context of the M-value, we partition the input courses by their semester of study. Then, we perform multiple iterations of clustering, proceeding chronologically through the input. At each iteration, we allow the radial tree to grow by a single level, while recursively clustering majors based upon the groupings created in the previous step. We call this "path projection" clustering because at each iteration the possible majors that can be clustered are limited to the subset created in the previous iteration. The result is that majors which have largely different freshman courses are less likely to be clustered together, and for two majors to be determined as similar in year three, say, they must have maintained a some semblance of similarity for both years one and two.

Path projection clustering produces quite different results than standard clustering. Looking at Figure 4.7a, one can immediately see both structural and coloring differences in this method. The distribution of similar majors to CS is also highly varied with these results. We can see from the visualization that the major closest to CS by M-value is Mathematics. One might expect for CS to be more closely related to Computer Engineering (CE) because of the large number of programming courses they share, but comparing the node-link diagrams for their C-value graphs reveals some key differences between the two curricula (see Figures 4.4 and 2.2). Although both majors contain core programming courses (such as COSC 302 and COSC 360), we can see that CS and CE students tend to take Physics, Mathematics, and other courses during different semesters. From 4.7b, once again we see that CS clusters with Music Education concentrations, although now they lie on even closer branches.

Opposite the CS region, on the left portion of the tree is the branch containing Honors Computer Science. This region now contains Computer and Electrical Engineering and their honors versions, along with the additions of Biomaterials, Nanomaterials, and Honors Psychology. Note the different levels where majors are paired. Because of the inherent temporal ordering of the path projection clustering, we can conclude that Computer and Electrical Engineering remain along a similar path for about two academic years longer than Honors CS and Honors Psychology. Such major to major connections are completely beyond the scope of the typical catalog offered to students. Through leveraging the data that universities already maintain, deeper understanding of curricula may be found both on course to course and major to major levels.

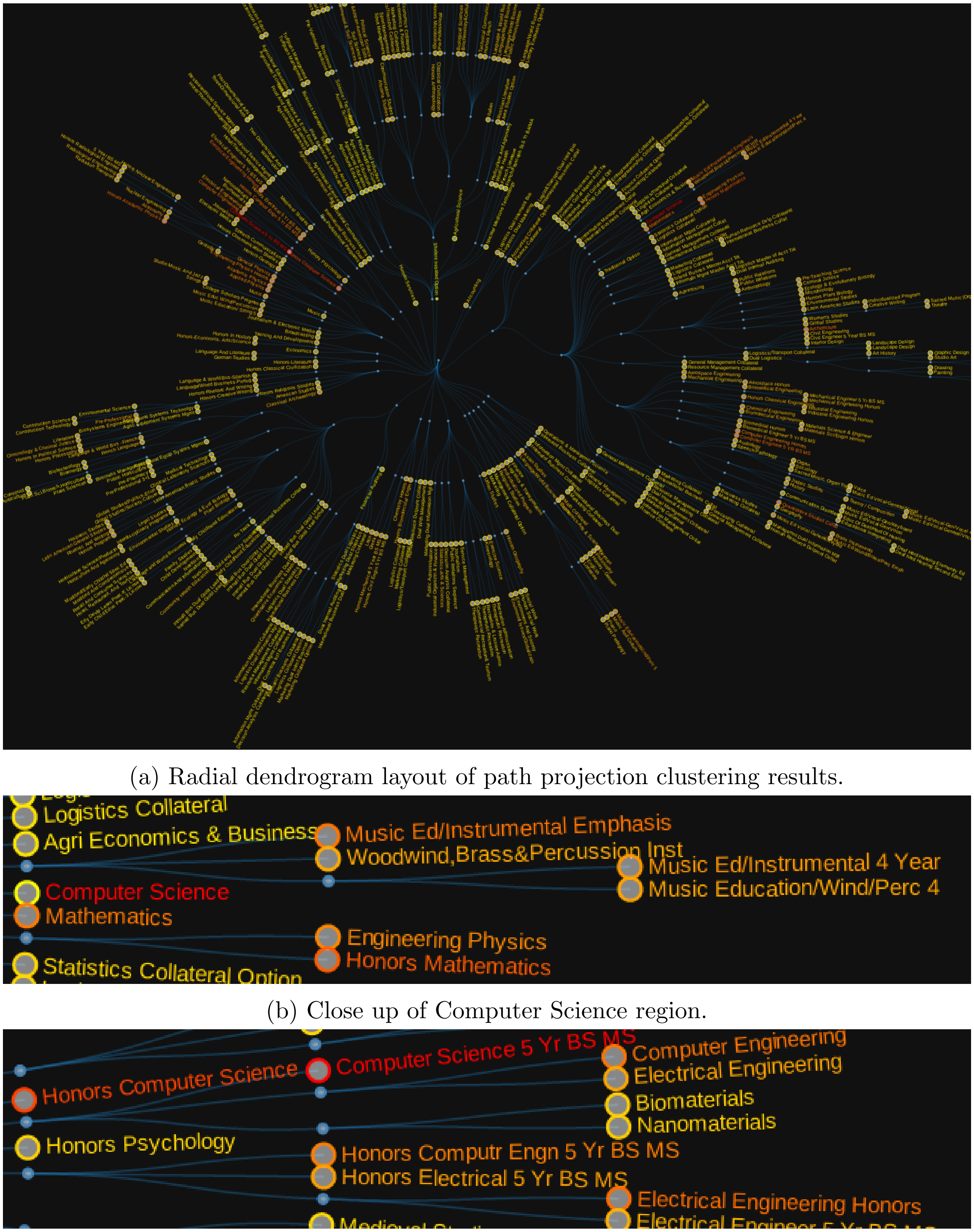


(a) Radial dendrogram layout of initial hierarchical clustering results.

(b) Close up of Computer Science region.

(c) Close up of Honors Computer Science region

Figure 4.6: Standard clustering results


Logistics Collateral
Agri Economics & Business
Music Ed/Instrumental Emphasis
Woodwind,Brass&Percussion Inst
Music Ed/Instrumental 4 Year
Music Education/Wind/Perc 4
Computer Science
Mathematics
Engineering Physics
Honors Mathematics
Statistics Collateral Option
Computer Science 5 Yr BS MS
Computer Engineering
Electrical Engineering
Honors Computer Science
Biomaterials
Nanomaterials
Honors Psychology
Honors Computr Engn 5 Yr BS MS
Honors Electrical 5 Yr BS MS
Electrical Engineering Honors

(a) Radial dendrogram layout of path projection clustering results.

(b) Close up of Computer Science region.

(c) Close up of Honors Computer Science region.

Figure 4.7: Path projection clustering results

Chapter 5

Concluding Remarks

5.1 Conclusion

This thesis presents data analysis performed on a novel dataset: university student records. We performed exploratory research into deriving new meaning from this previously studied data source. Along the way, we created a framework for analyzing and visualizing categorical data with temporal features and a parallel pipeline for performing statistical analysis to facilitate interactivity and analysis with the data. From this analysis, we created two measurements of similarity within the university curricula: the C and M values. Using the derived graphs of similarity values, we leveraged and extended upon current visualization techniques for understanding graphs to reveal previously unknown structure within the university curricula. We also performed hierarchical clustering to determine not only how majors within a university are related to each other, but to reveal previously unknown structure within these relationships. Our temporal partitioning and recursive, iterative clustering allowed us to view how closely majors share course loads from first year on. Overall, we provided an new way to view and interact with this dataset, revealing hidden trends along the way. Ultimately, the approaches taken in this thesis are generic enough so that they may be applied to any pipeline where routine, temporally consistent measurements are taken along the way.

5.2 Future Work

Approaches in this work were taken with the driving goal of exploratory and curiosity based research in mind. With this preliminary work set on a solid foundation, expansions from exploratory visualizations to concise, predictive functionality would be the next step. With the vast amount of correlation information calculated, using machine learning to provide some form of schedule builder, which could flexibly order courses to provide the greatest probability of student success, would be a worthy extension. Easily providing students with a list of majors they are moving closer towards for possible graduation and which majors they can switch to, with the least amount of wasted credit hours, is another step in extending this work. Opening our work for other universities to explore will benefit students, faculty, staff, and administrators, as well as bring the collegiate schedule building experience into the modern era of data-driven improvements to everyday life.

Bibliography

[1] Fabian Bendix, Robert Kosara, and Helwig Hauser. Parallel sets: Visual analysis of categorical data. In *Information Visualization, 2005. INFOVIS 2005. IEEE Symposium on*, pages 133–140. IEEE, 2005. 9

[2] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006. 18

[3] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17 (12):2301–2309, 2011. 7, 22

[4] Marcia Roe Clark. Negotiating the freshman year: Challenges and strategies among first-year college students. *Journal of College Student Development*, 46 (3):296–316, 2005. 1

[5] Greg N Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM Journal on Computing*, 14(4):781–798, 1985. 8

[6] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991. 7

[7] Mohammad Ghoniem, J Fekete, and Philippe Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Information Visualization, 2004. INFOVIS 2004. IEEE Symposium on*, pages 17–24. IEEE, 2004. 6, 8, 26

[8] Ivan Herman, Guy Melançon, and M Scott Marshall. Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1):24–43, 2000. 7, 10

[9] Danny Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, 2006. 7

[10] U.S. Immigration and Customs Enforcement. Stem-designated degree program list. http://www.ice.gov/doclib/sevis/pdf/stem-list.pdf, 2012. 26

[11] Vince Kellen J. David Hardison. Improving student success: Using groundbreaking analytics and fast data to improve student retention. 2013. URL http://www.educause.edu/annual-conference/2013/. EDUCAUSE Annual Conference. 4

[12] Adam Jacobs. The pathologies of big data. *Communications of the ACM*, 52 (8):36–44, 2009. 2

[13] Brian Johnson and Ben Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Visualization, 1991. Visualization'91, Proceedings., IEEE Conference on*, pages 284–291. IEEE, 1991.

[14] George D Kuh, Jillian Kinzie, John H Schuh, and Elizabeth J Whitt. *Student success in college: Creating conditions that matter.* John Wiley & Sons, 2010. 4

[15] Richard Light. Strengthening colleges and universities. Harvard Assessment Seminars (Forum for the Future of Higher Education, EDUCASE, Aspen, Colorado, 2006). 1

[16] Michael J Lutz, James R Vallino, Kenn Martinez, and Daniel E Krutz. Instilling a software engineering mindset through freshman seminar. In *2012 Frontiers in Education Conference Proceedings*, pages 1–6. IEEE, 2012.

[17] Riccardo Mazza and Vania Dimitrova. Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 154–161. ACM, 2004. 5

[18] Riccardo Mazza and Vania Dimitrova. Generation of graphical representations of student tracking data in course management systems. In *Information Visualisation, 2005. Proceedings. Ninth International Conference on*, pages 253–258. IEEE, 2005. 5

[19] University of Tennessee Office of the University Registrar. uTrack Overview, 2014. URL `http://registrar.tennessee.edu/utrack/`. 4

[20] Urko Rueda, Mikel Larrañaga, Mikel Kerejeta, Jon A Elorriaga, and Ana Arruarte. Visualizing student data in a real teaching context by means of concept maps. In *International Conference on Knowledge Management I-Know*, 2005. 5

[21] Chad A Steed, Patrick J Fitzpatrick, TJ Jankun-Kelly, Amber N Yancey, and J Edward Swan II. An interactive parallel coordinates technique applied to a tropical cyclone climate analysis. *Computers & Geosciences*, 35(7):1529–1539, 2009. 9

[22] Ellen M Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6): 465–476, 1986. 18

[23] Leland Wilkinson and Michael Friendly. The history of the cluster heat map. *The American Statistician*, 63(2), 2009. 15

Vita

Blaise DeCotes was born in Knoxville, Tennessee on June 7th, 1990. He graduated from Knoxville Catholic High School in 2008, and continued his education by attending the University of Tennessee in Knoxville. He received his Bachelor of Science degree in Computer Science, with a second major in Mathematics, in May of 2012. He worked as an undergraduate research assistant for the Joint Institute for Computational Sciences during his fourth year of undergraduate studies. This experience inspired him to pursue graduate-level education. He received his Master of Science degree in Computer Science in August of 2014.