# CourseCompass Final Report

## Introduction

[CourseCompass](https://github.com/Parth099/Math-Modeling-Final-Project-Course-Model) is an Agent-Based Model (ABM) based on how students progress across a learning curriculum that contains prerequisites. In this model, each student is represented as an individual agent that interacts with the learning environment and other agents. The ABM takes into account the prerequisite knowledge required for each class in the curriculum and tracks each student's progress through the curriculum based on their prior knowledge. Each timestep in this model represents a semester. This means that they are assigned classes based on what they have taken before and subsequently assigned grades based on what they have achieved before. CourseCompass also takes into account individual class sizes as in real institutions class sizes may vary as a student moves through courses. This introduces an agent-agent interaction where a student may not get a class they need even if they have completed all the stated requirements because the class is full. In terms of modeling ability, the internal course model can model any “and” relationship where a set of requirements *must* be met to take a class. However, only elementary “or” relations can be modeled where *x* out of *y* items must be completed to take a class.

The reason this topic was interesting to me was initially because of the graph theory involved. If we look at this mathematically, the topological sorting of a directed acyclic graph provides an ordering of the graph where each node precedes each next node in the graph. In other words, it gives an ordering where if node A appears before node B, A is a prerequisite of B or can be visited before B. Later, I remembered that temple has a ‘fly-in-4’ program where they guarantee your graduation in 4 years by simply checking in with you each semester and talking about your grades and class selection which is an interesting promise. I chose this curriculum since I know it the best and has the course-course relationships most curriculums contain. There is currently no research done on this topic based on queries done on [Google Scholar](https://scholar.google.com/) and [Elicit](https://elicit.org/).

The current course data loaded onto the model is the Temple University Computer Science Curriculum. In this curriculum, not all students have to take the same classes to graduate outside of the major classes. For example, a student must pass six CIS electives which can vary per student.

## Methods

In this model the agents are individual students while the environment is the plane of courses and prerequisite relationships between courses. Each agent has the following attributes:

* Course Plan: A plan to all classes required to graduate in a certain order. Between students the size of the course plan can vary since a student is free to *choose* whether to meet the minimum requirements or go beyond them. The assumption of students taking more than what's required is based on real life observations. For example, if a student feels that their fundamentals are weak they will take both of the intro courses rather than just the singular required one.
* Semester: Keep tracking of what semester each agent is in
* has\_taken, is\_taking, has+failed: Keep tracking of what each agent has taken, is taking, and has failed to make grading and registration decisions.
* Grades: grades a student has gotten per class to make future grading decisions
* credit\_count: used to limit the number of classes a student can take per semester.

The school-based environment has courses which have the expected attributes: code, name, and credit number. However, the only attributes that affect agents are:

* Prerequisites: Classes referenced in the `Prerequisites` set have an ‘and’ relation where an agent must complete each prior class to get a chance to enroll in this one.
* Requirements: These represent the ‘or’ relationships. A student is given the choice to complete *n* classes in these lists to fulfill requirements. For example, the course directory contains 6 electives yet students only need to complete any 3. However, some agents will complete more than 3.
* Class-size: This attribute is designed to induce agent-agent interactions where agents that apply for a class get it first to model real life situations where students are limited not by prerequisites, but class sizes.

There were a few considerations to make for this model. First, the grading scheme. Initial grading was based on a ‘random walk’. However, this is not like real grading assignments. It should seem that students that perform well in similar classes should perform the same way in the current classes they are taking. Thus, students now receive grades based on their past performances. Initially each student gets a grade defined on a normal distribution () if there is no history, however if the class does have similar classes (prerequisites) the mean of the normal distribution is shifted to the mean of their grades in the prerequisite classes. Agents tend to try to take as many classes as they can up to the semesterly, credit limit which is 18.

Another challenge was generating the course path for students. The initial idea was to use a topological sort but it had the issue of including each class in each category, but that disabled the ‘or’ feature in prerequisite chaining. I ended up using a bucketing system where I would create buckets of required classes per requirement and then later combine buckets to create an agent’s course plan. For example, in the electives bucket I would randomly assign 3 electives and then randomly assign some agents more as this represents agents who wish to learn more.

## Results and discussion

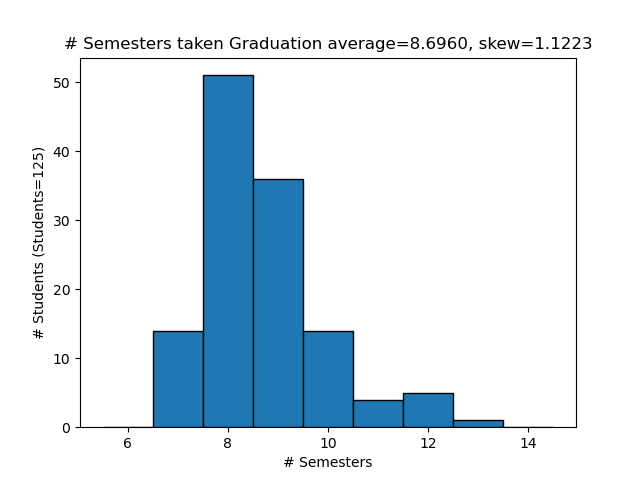
### Grading: Random Grades Vs. History-Based Grading

The model supports both grading schemes based on a flag being set.

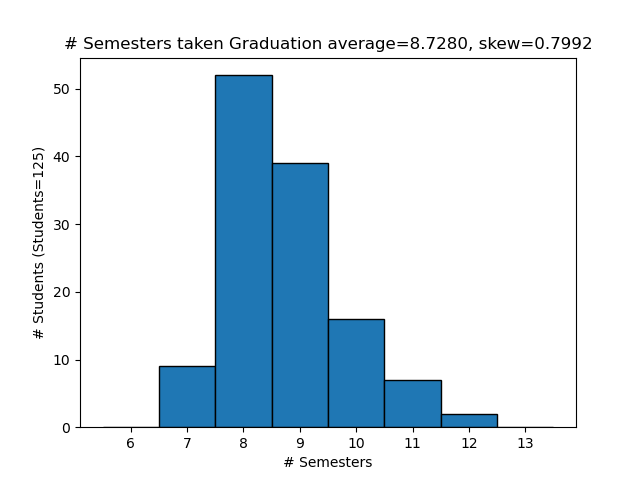
Over a 1000 independent trials using 125 agents:

|  | History Based Grading | Random Grading |
| --- | --- | --- |
| Average Skew On Semesters to Graduate | 1.264699 | 0.679456 |
| AverageOn Semesters to Graduate | 8.764512 | 8.779744 |

Based on those trials it seems that the skew is nearly double when using history based grading yet the number of semesters to graduate is about the same. This can be explained by compounding behavior.



Using history-based grading it is more probable to have agents graduate early or very late since their initial performance determines their future pathway. This means that if an agent does poorly initially they will do poorly later since grades are determined by earlier performances. On the left are sample histogram distributions on how many semesters it took for 125 students to finish out the curriculum using history based grading. Notice that these histograms have more mass on the more extreme values.

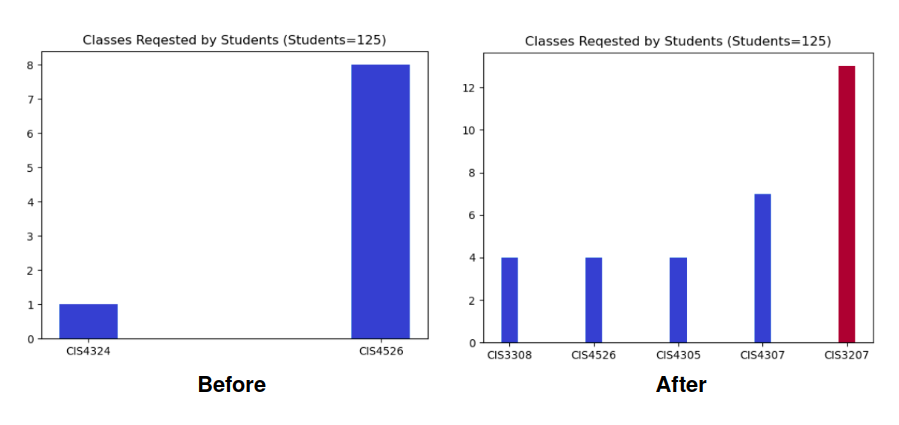


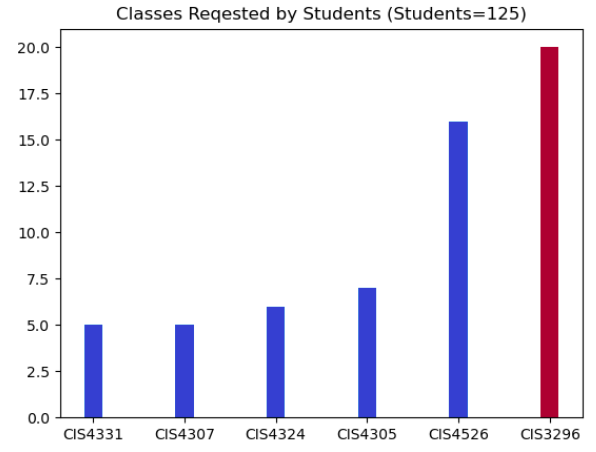
One the right are sample histograms where random grades are assigned and it is easy to notice that the distribution looks like a normal distribution where most students take 8 or 9 semesters to graduate.

While there is no distribution for grading it seems for agent based model, history-based grading is a more realistic strategy since it can capture higher and lower performing students better than random grading.

### The Effect of Class Sizes on Subsequent Classes

In many situations, students have the prerequisites completed yet are required to waitlist since the university did not foresee the influx of students wanting to take a certain class. This emergent behavior from the agents is what I have labeled as ‘bottlenecking’, which results from the low number of seats in a popular class. For the sake of example, I will reduce the number of seats in a required class, CIS3207, from 100 to 50.



Notice that due to randomness some other classes appeared but the class that was reduced shows up as the primary bottleneck. This can help universities understand where they need to allocate more resources, but that is not the whole story. If the initial conditions of this model had CIS3207 at 50 seats, classes that require that class would see fewer students and the university would not allocate many resources for that class initially. However, if the university sees CIS3207 as a bottleneck down the line and increases the seats from 50 to 100, other classes would be affected that were initially downsized. The chart on the left shows the effect of increasing the seats in CIS3207 while not increasing the seats in classes that consider it a prerequisite. Evidently, when considering altering the number of seats in a course, a university should also consider how it will affect other classes.

## Conclusion

This project studied course flow with agents that take courses based on their previous course history and prior grades. It studied the effects of different grading schemes by analyzing them under a monte carlo simulation. The results of one thousand course simulations showed that history-based grading induces a greater skew in the number of semesters to graduate due to the effect of compounding success or failure. Students that received better grades earlier on were able to maintain better grades for the entirety of the simulation. The project was also used to examine the emergent effect of changing class sizes. The results showed that the effects of altering a class size depend on the children of the node changed (classes that require *it* as a prerequisite).

One of the primary things I learned in this project is the chained effect of changes induced in a graph. In the discussion above we changed a property of a node that was a predecessor to another node and it had unintended consequences. To solve the inadequate class size of one class, denoted by the waitlist requests by agents, the class size was increased yet it caused the bottleneck issue to just move to the next class in line. Due to this example, I learned to consider the node’s children when changing the properties of the child. This ultimately led me to do more research on the [Maximal Flow Problem](https://en.wikipedia.org/wiki/Maximum_flow_problem), which attempts to find a maximal flow over a network. In this case, it will highlight the nodes that cause the biggest bottlenecks in the course follow deterministically. Unfortunately, Maximal Flow methods were not applied since the graph is not fully connected and cannot properly represent ‘or’ prerequisite relationships on a graph level.

The next step for this model is to implement more complex agent behavior. For example, I was considering adding a 2 step timestep for each semester. The first step of the timestep would assign classes and a midterm grade. Based on this midterm grade, students could choose to drop the class or keep it based on an agent's risk level. However, this behavior would be more complex to program and lead to similar results since a drop is programmatically equivalent to a failure. In terms of feedback received from Dr.Seibold, I agreed that a ‘retake’ agent behavior is necessary for a system where history-based grading is involved. In a real system, a student may retake a class to improve their GPA and thus accrue more knowledge in a class. This can lead to an initial slowdown in progress in a history-based grading model, but accelerate progress once the student gets better grades. This would be the change I would make if I were to rebuild this model.

## 

## Bibliography

Wu, Gang. “Topological Sort of Directed Acyclic Graph | Baeldung on Computer Science.” *Www.baeldung.com*, 14 July 2020, [www.baeldung.com/cs/dag-topological-sort](http://www.baeldung.com/cs/dag-topological-sort).

“Exams - What Does an Optimal Grading Distribution Look Like?” *Academia Stack Exchange*, academia.stackexchange.com/questions/61038/what-does-an-optimal-grading-distribution-look-like. Accessed 5 May 2023.

‌

Wikipedia Contributors. “Maximum Flow Problem.” *Wikipedia*, Wikimedia Foundation, 8 Feb. 2019, en.wikipedia.org/wiki/Maximum\_flow\_problem.

‌

‌