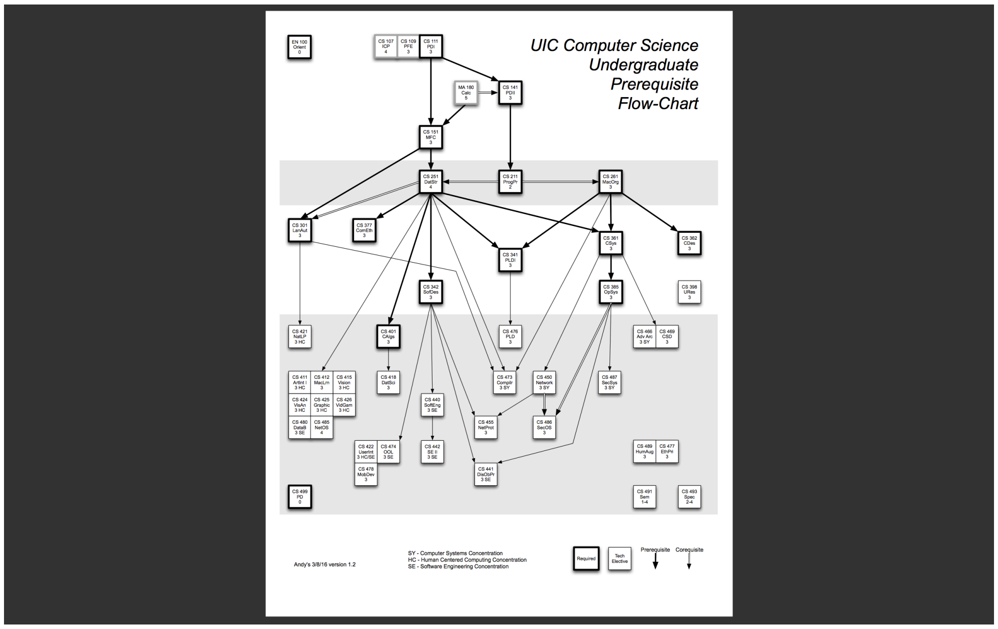
Pranav Rajiv
CS 524 Project
9072019053

Cost Minimization

A student wants to graduate with a Computer Science degree from University of Illinois at Chicago. For graduating UIC requires the student to take a certain number computer science classes. For taking these specific classes, there are multiple prerequisite options. Each class has a certain number of credits and the student is charged a semester fee, which is $15000, and the student is not allowed to take over 18 credits in the semester. In my program I have introduced many things learned in class like the Big M notations, Network Flow and Trade-off constrains. My model is constructed using mixed integer programming.

The data that I used in this project is from a flowchart that UIC offers for its undergraduate computer science students. This flowchart contains class credits and its prerequisites. I am getting the time and semester availability of each class from the UIC's class registration link that is available to current students and previously enrolled students of UIC. Figure 1 below is a flowchart of the CS classes in UIC

Figure1



I met the prerequisites for each class in my program by using the network flow equations covered in the network flow section in lecture. For that I have created a set of arcs, which determines what are the prerequisites of each class. These arcs are one directional. The code snippet below makes sure that a class is taken only if its prerequisites are met.

```
flow_balance0(class,class1,sem)$(Arcs(class,class1))..
sum(time,x(class1,time,sem))=l= sum(time,sum(sem1$(Ord(sem1)<Ord(sem)),x(class,time,sem1)));
```

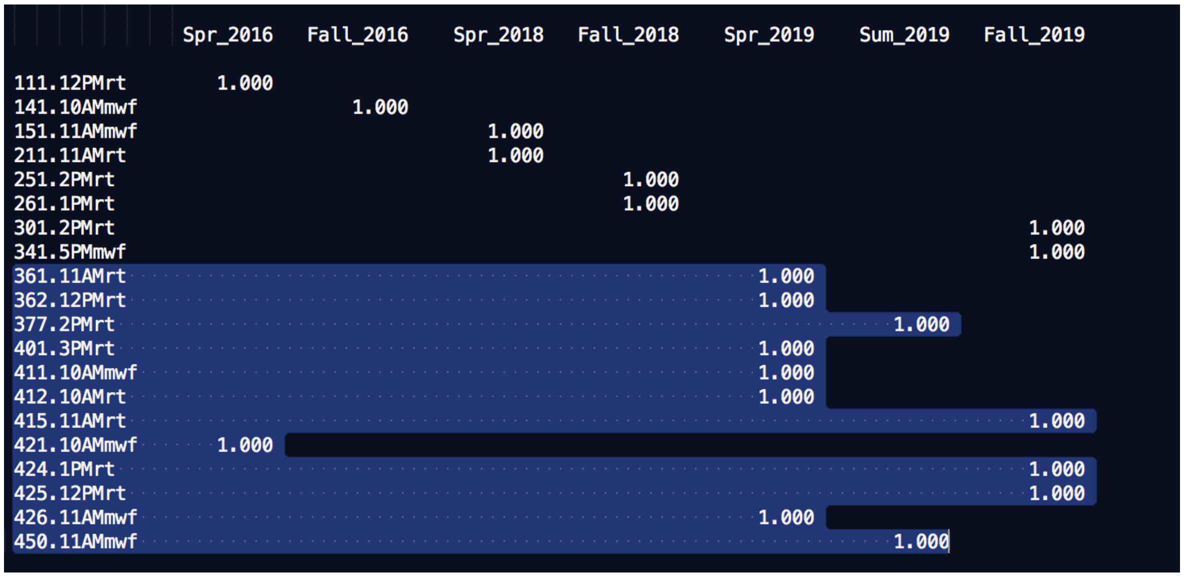
In the code snipped below I have introduced the concept of Big M notation that was discussed in class. The value of Big M in the code below is 6. This makes sense because most of the CS classes are 3 credits and as six times 3 is 18, it's the maximum number of credits a student can take in a semester . So the sum of the classes taken in each semester would not exceed 6.

```
crseTke(sem)..
sum((class,time),x(class,time,sem)) =L= 6 * semesterTaken(sem);
```

In my program I have introduced many constrains. One of these constrains limits a student from taking a class more than once. Another constrain that makes sure that the student doesn't take 2 classes at the same time in the same semester. I have also introduced constrains with prevents a student from taking more than 18 credits in a semester. The program also has constrains which tells that the student can't graduate without 60 credits.

I have 2 solve statements in my project. The first solve statement finds what are the best classes the student can take without considering how difficult those classes are. The second solve statement introduces the difficulty parameter which has information on a 1-5 scale, on how difficult a specific class is in UIC. The pictures below are snippets of the best classes to take with and without the difficulty constrain.

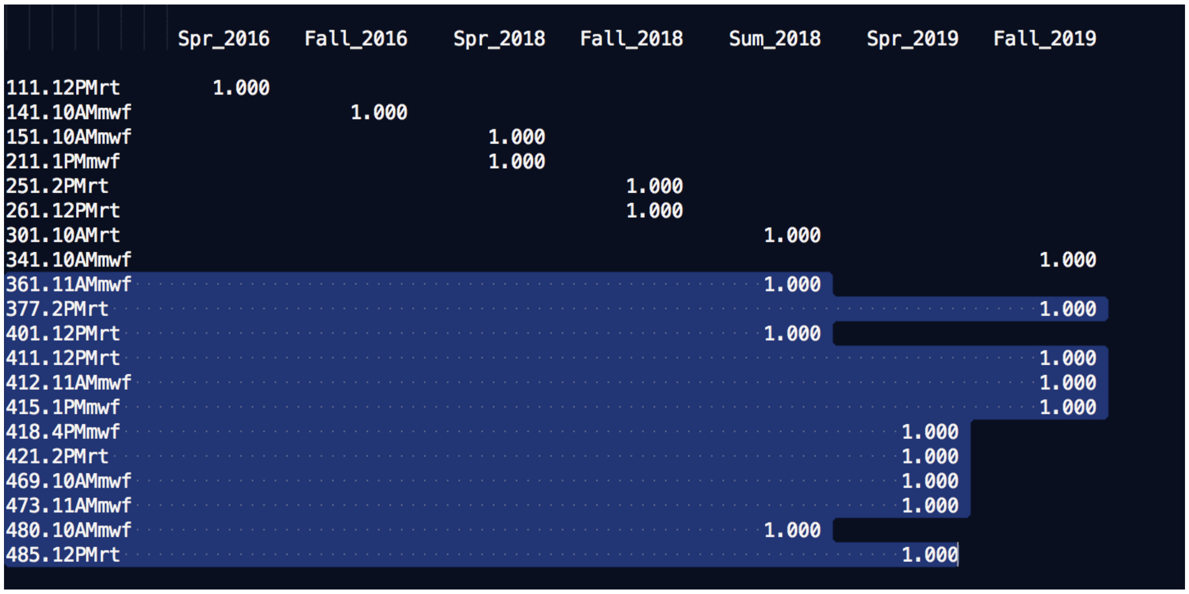
Figure 2



```
                Spr_2016   Fall_2016    Spr_2018   Fall_2018    Spr_2019    Sum_2019   Fall_2019

111.12PMrt         1.000
141.10AMmwf                    1.000
151.11AMmwf                                1.000
211.11AMrt                                 1.000
251.2PMrt                                              1.000
261.1PMrt                                              1.000
301.2PMrt                                                                                  1.000
341.5PMmwf                                                                                 1.000
361.11AMrt                                                         1.000
362.12PMrt                                                         1.000
377.2PMrt                                                                      1.000
401.3PMrt                                                          1.000
411.10AMmwf                                                        1.000
412.10AMrt                                                         1.000
415.11AMrt                                                                                 1.000
421.10AMmwf        1.000
424.1PMrt                                                                                  1.000
425.12PMrt                                                                                 1.000
426.11AMmwf                                                        1.000
450.11AMmwf                                                                    1.000
```

Figure 3



```
              Spr_2016  Fall_2016  Spr_2018  Fall_2018  Sum_2018  Spr_2019  Fall_2019
111.12PMrt       1.000
141.10AMmwf                 1.000
151.10AMmwf                           1.000
211.1PMmwf                            1.000
251.2PMrt                                        1.000
261.12PMrt                                       1.000
301.10AMrt                                                 1.000
341.10AMmwf                                                                     1.000
361.11AMmwf                                                1.000
377.2PMrt                                                                       1.000
401.12PMrt                                                 1.000
411.12PMrt                                                                      1.000
412.11AMmwf                                                                     1.000
415.1PMmwf                                                                      1.000
418.4PMmwf                                                           1.000
421.2PMrt                                                            1.000
469.10AMmwf                                                          1.000
473.11AMmwf                                                          1.000
480.10AMmwf                                                1.000
485.12PMrt                                                           1.000
```

The figure 2 is the output of the best class to take without considering the difficult constrain. The program just decides what is the best path to follow and take classes so that the student can complete his/her 60 credits requirements.

The figure 3 is the output of the best classes to take when difficulty constrain is introduced. Based on this constrain the program reconsider the best classes the student can take that is the program tries to take easy classes and also thinks about if its cost effective so that it increase the student's chance of getting a better grade in those classes without paying too much price for it.

When we compare the outputs form figure 2 and figure 3 we see a diffident class schedule the student is recommend to take in each case. In both figure 2 and figure 3 we see that till CS 361 it's the same schedule with and without the difficulty constrain. After CS 361, in figure 2 it's recommender that the student takes CS 362. But in figure 3 after CS 361 it's recommended to take CS 377. The reason why there is a change in output is because CS 362 has a difficulty level of '5' so the model recommended to skip that class. This reason made the program skip to CS 377 instead of taking CS 362. It is the additional trade off constrain which does that.

The program has optimized to take all the CS class in 7 semesters instead of 12 semesters (4 years), which saves the student a lot of money. Based on the program, the final cost for the student to meet the computer science requirements for a CS degree from UIC is $ 105,000. The output makes sense because it takes the student 7 semesters, starting Spring 2016 and ending in Fall 2019,leaving some semesters in between, to meet all the Computer Science requirements. Since each semester would cost the student $15,000, seven times fifteen thousand is $105,000.

This project creates a model that says which are the best computer science classes to take in UIC. I have introduced many constrains in my project to make it more real world friendly. With this project I was able to implement many concepts that I learned in class and above all understand how an actual optimization program looks like. I would like to thank the University of Illinois at Chicago Computer Science department for providing its undergraduates with the CS flowchart and also for letting me use the flowchart.

References: https://www.cs.uic.edu/~grad/CS_Flowchart_03082016.pdf