MATH 381 Winter 2020 - Project 1 Paper

(Group 2) Daehyun Chung, Erik Huang, Quynh Huynh, Tianhao Jiang

Instructor: Matthew Conroy

August 15, 2020

1 Introduction

Undergraduate students at the University of Washington (UW) follow different degree paths in order to fulfill general education requirements and specific major courses. Typically students would then graduate within 4 years while satisfying all requirements, they either pick their own classes or follow advisors' suggestions while building their quarterly class schedule. In this paper, we will use Integer Programming to explore the optimal plan that uses the least number of quarters for a major-specific student to graduate. The solution from our student scheduling problem will produce the student's plan spanning the optimized minimum number of quarters. The plan will satisfy all kinds of major requirements, restrained by credit-limit per quarter and course prerequisites.

2 Background

We picked graduation as our main topic for the project because, as juniors and seniors, we all wondered: "What would the best fitting schedule for graduation be for a certain science-oriented degree at the University of Washington?" An easy answer would be to take as many credits as possible every quarter, but we want to create an optimal proper schedule by using linear programming with certain constraints per quarter (will be mentioned later). A few of these constraints include limiting to 15 credits per quarter and excluding seat availability from consideration.

We have specifically decided on student schedule who plans to major in Computer Science (CSE) for many reasons. First, we want to look into the amount of time it takes for a STEM-focused student to graduate at the UW. Second, the intensity of the CSE coursework is well-known, with a good balance on liberal arts and general education requirements with greater flexibility for the student's upper-division classwork. Since our university is research-focused, the expectations for student performance is well-rounded throughout all fields of study. The CSE degree's major requirement also has the highest number of credits for the student to satisfy (90 credits). Thus, looking into the CSE diverse coursework would allow us to better understand the general STEM-student at the UW.

Scheduling has been a prominent topic in optimization problems with many variations from job shop scheduling to nurse scheduling. The main goal of these problems is to optimize shifts for workers in a firm under a set of constraints. These types of problems have long roots dating back to the early 1950s where researchers worked on the several solutions using of barcharts, milestones, and linear programming.

Academic scheduling problems later emerged in early 1960s with authors' main focus into the use of school timetables and teachers' availability. In 1961, Appleby, J.S.(1961) wrote a computer program for producing a timetable for a medium-sized school using computer logic to fill up a blank timetable by one line entry at a time where the next line to be entered is chosen by how difficult it is

to fit in classes. This approach differs from our group project since the authors used trial-and-error method where the entry to fit a specific class is proven best out of all the remaining entries (which means all previous entries had to be tested before reaching the destination entry). The computer that the authors were using took 1.5 hours to complete a full schedule, which was proven much faster compared to the 100-hour work that the schoolmaster would usually have to spend.

Salazar and Oakford (1974) gave us a different perspective on solving for school schedules. The authors develop a general algorithm to formulate a non-directed graph and determine an optimal school exam schedule from the largest strongly connected subgraph. Each nodes are the exams to be scheduled and the arcs' binary weights determine whether the two exams can be hosted concurrently. The size of this strongly connected component is the entire graph's lower limit of subsets where two members of the same subset are connected by an arc. From this subset, the school can determine what periods to host the exams from a given number of periods that can be used for scheduling. This is very different from our group's approach since we only consider the linear model to ensure prerequisites are applied. Thus, the authors were looking into exam schedule solutions while we are working for a personalized student's class schedule.

The study on the school schedule problem gradually opened up to individualize students' schedules. Feldman and Golumbic (1990) compared different optimization algorithms for finding an optimal study schedule. They set up the study by using 1000 different problem set-ups and using multiple methods of optimization methods, then they rank the performances based on how well the algorithms could handle the constraints (how few constraints were relaxed until they reach a solution). Their result concluded that the offering ordering algorithm (a regular forward checking system where criteria of selection is updated after each sort of classes) performed best. The study was interesting since the authors aim to study the performance of algorithms where put an emphasis on the constraints that are subject to the student's preferences and requirements. Our project also aims to solve an individual student's schedule, but we specifically computed unique solutions that applies to all of the constraints.

In recent years, Guo et al. (2013) presents an integer programming model that solves for a residency schedule for medical school graduates. Their integer program takes in constraints subject to the specific program demands, educational requirements, and the rotation placement and period, and developed a single-year schedule for the students. Though their approach in satisfying different requirements is very similar to our group project since they used interger programming that utilized binary variables, our schedule solution actually spans for 4 years for the UW student.

Gulera and Lal (2019) worked on the school schedule problem from a very interesting aspect: maximizes the number of backlog courses offered to college juniors that fit into their existing timetable. The goal is for the students to be able to retake courses that they did not passed (called 'backlog' courses) in a subsequent quarter that do not clash with their regular courses (including lectures and

quiz sections). The solution method that the authors used was a multi-level optimization model, but the authors later decided that dynamic programming was the better platform for computation. The study focused on a variation of the school scheduling problem but with goals to maximize classes that should be added to help students to graduate on time. The paper differs from ours since we work on a minimization objective function.

3 The model itself

IP formulation

We attempted to solve this schedule finding problem with Integer Programming. Our goal is to optimize course schedules for a student completing a B.S degree in Computer Science to graduate in the fewest number of quarters possible. This problem can be modeled as an Integer Programming problem because decisions of whether to take a certain course at a certain time period can be represented as binary variables. In addition, our primary goal is to minimize the number of quarters required to graduate, which is suitable for the objective function. The feasibility of this problem is all the possibilities of schedule plans that lead to a successful graduation. The LP solver program can be used to perform the optimization and obtain the plan with the minimum amount of quarters required.

Variables	Range	Description
y_q	$y_q \in \{0,1\}$, $q = 1, ..., 12$	$y_q = 1$ iff student is enrolled in quarter q
x_{qc}	$x_{qc} \in \{0,1\}$, $q = 1, ..., 12$, $c \in$ all courses	$x_{qc} = 1$ iff student is enrolled in course c in quarter q
z_c	$z_c \in \{0,1\}$, $c \in$ all courses	$z_c = 1$ iff student is enrolled in course c before graduation
o_{qc}	$o_{qc} \in \{0,1\}$, $q = 1, ..., 12$, $c \in$ all courses	$o_{qc} = 1$ iff course c is offered in quarter q
cr_c	$cr_c \in \{1,2,3,4,5\}$	$cr_c = 5$ iff course c gives 5 credits
$k_{crtp,c}$	$k_{crtp,c} \in \{0,1\}$, $crtp \in$ all credit types, $c \in$ all courses	$k_{crtp,c} = 1$ iff course c offers $crtp$ credit type
max_q	$max_q \in \{12, ..., 18\}$	$max_q = 15$ iff student can take a maximum of 15 credits in quarter q
req_{crtp}	$crtp \in$ all credit types, $\text{req}_{crtp} \in 1, 2, ...$	$req_{crtp} = 20$ iff graduation requires 20 credits of $crtp$ credit type
req	$req = 180 -$ AP_credits	The minimum number of total credits required to graduate is 180 minus AP credits

This chart contains all variables presented in our model. Among them, only x_{qc}, y_q and z_c will be seen in the LP input file. The reason is that the value of other variables can be pre-determined from the data we collected from the department website. Our source of data comes from official websites from UW: CSE Time Schedules, COMPUTER SCIENCE AND ENGINEERING COURSE OFFERINGS, Computer Engineering Graduation Requirements.("COLLEGE OF ENGINEERING,"2020)(CSE Teaching Schedule 2019-2020,2020)("Degree Requirements,"2020) The first site provides course offering information. The second site provides course prerequisite information. The third site provides the

graduation requirement.

Then, our IP is presented as the following:

$$\text{Minimize } \sum_{q=1}^{12} y_q \text{ subject to}$$

$$\begin{cases} \frac{1}{20}\sum_c x_{qc} \le y_q, \quad q = 1, ..., 12 \\ \sum_c x_{qc} \ge y_q, \quad q = 1, ..., 12 \end{cases} \tag{1}$$

$$\begin{cases} \frac{1}{20}\sum_{q=1}^{12} x_{qc} \le z_c \text{ for all courses} \\ \sum_{q=1}^{12} x_{qc} \ge z_c \text{ for all courses} \end{cases} \tag{2}$$

$$y_q \le y_{q-1}, \quad q = 2, ..., 12 \tag{3}$$

$$x_{qc} \le o_{qc}, \quad q = 1, ..., 12 \tag{4}$$

$$\sum_{q=1}^{12} x_{qc} \le 1 \text{ for all } c \in \text{(all courses)} \tag{5}$$

$$\sum_{q'=1}^{q-1}(x_{q'c2} + x_{q'c3}) \ge x_{qc1}, \text{ if c2 or c3 is the prerequisite of c1}, q \ne 1 \tag{6}$$

$$\sum_c x_{qc} cr_c \le max_q, \quad q = 1, ..., 12 \tag{7}$$

$$\begin{cases} \sum_c z_c k_{crtp,c} cr_c \ge req_{crtp} \text{ for all } crtp \in \text{(all credit types)} \\ \sum_c z_c cr_c \ge req \end{cases} \tag{8}$$

$$z_c = 1 \text{ for all c} \in \text{(CSE core courses)} \tag{9}$$

$$x_{qc} = 0 \text{ for all c} \in \text{(CSE 400 level courses)},$$
$$q = 1, ..., int(7 - (\text{AP\_credit}/10)) \tag{10}$$

$$\begin{cases} \sum_c x_{qc} k_{crtp,c} \le 3, \quad q = 1, ..., 12, \quad crtp = \text{"CSE"} \\ \sum_c x_{qc} k_{crtp,c} \le 5, \quad q = 1, ..., 12, \text{ for all } crtp \in \text{(all credit types)} \end{cases} \tag{11}$$

Constraint (1) requires a quarter q to be considered as taken if and only if the student takes any classes in the quarter q. This constraint contains 2 inequalities. The first one ensures that a quarter must be taken if there's any classes planned in that quarter. We need to sum up all the x_{qc} for each specific quarter and check if the sum is greater than 0. We also need to make the left side of the

inequality less than or equal to 1 in order for it to handle all cases. So we decided to divide the sum by an integer 20 because a student can only take at max 15-18 credits per quarter. Dividing the sum by 20 will always make it less than 1. The second inequality ensures that a quarter cannot be taken if there's no classes planned for that quarter.

Constraint (2) requires a course c to be marked as taken if and only if the student has taken the course c during any quarter before the graduation. This constraint also contains 2 inequalities. As explained in the chart above, $z_c = 1$ means that course c is taken before graduation. The first inequality ensures that a course c must be marked as taken if there's any class planned in that quarter. Here we need to sum up all the x_{qc} for each specific course and check if the sum is greater than 0. We also need to make the left side of the inequality less than or equal to 1 in order for it to handle all cases. In this case we divide the sum by an integer 20 because there are only 12 quarters in total. Any integer greater or equal to 12 can make the sum less than or equal to 1. The second inequality ensures that a course cannot be marked as taken if it is not taken during any quarter.

Constraint (3) requires all quarters to be used in order. This constraint is necessary because we consider the index of all 12 quarters in chronological order. This prohibits the probability that the program uses a quarter in the future before it fills the current quarter.

Constraint (4) states that a course c can be taken in quarter q if it is offered in the quarter q. This constraint aims to make the problem solution more realistic since in reality not all courses are offered during every quarter. The variable o_{qc} is not shown in the LP input file. Instead, the program writes 0 or 1 directly because the course offering information is already loaded into the program. However, in our main solution, we choose to consider all courses to be offered in every quarter because there's a lot of ambiguity in the official course offering website that troubles the program to load it.

Constraints (5) states that every distinct course c can only be taken once among all available quarters. The reason why this constraint is added is to prevent the program from picking out good courses and repeatedly taking them until graduation. Realistically, most students only take each course once. In this problem, we also chose not to consider the retake of any courses.

Constraint (6) states that a course $c1$ can only be taken in quarter q if its prerequisites have been fulfilled before that quarter.

Constraint (7) states that the number of credits taken in any quarter q cannot exceed max_q, the maximum number of credits allow in that quarter. The allow range of max_q set is from 12 to 18 to conform to the UW policies. In our main program, we set $max_q = 15$ for all quarters to ensure that the student is not overloaded.

We also considered whether to include the constraint $\sum_c x_{qc} cr_c >= 12$ to implement the policy

that students must take at least 12 credits each quarter. We found this constraint to be difficult to implement since we needed to consider the special case that there is no such requirement during the quarter of graduation. We also found the constraint to be unnecessary since the solutions generated from our existing constraints always have already fulfill this constraint.

Constraint (8) states that the number of credits taken of a given credit type must be greater than or equal to the it's corresponding graduation requirement. The second line of Constraint (8) is a special case of the constraint which ensures that the total number of credits taken, disregarding the credit type, must be greater than or equal to the variable req.

Constraint (9) ensures that all CSE core courses are taken before graduation. This is a requirement of graduation with a CSE degree.

Constraint (10) states that the student cannot take CSE 400 level courses in the first "int(7 - (AP_credit / 10))" quarters. This constraint takes into consideration the difficulty of CSE courses. CSE 400 level courses are generally difficult, so students don't generally take them until their sophomore or junior year when they enter the major. However, a student may still take these courses earlier depending on their own situation.

We decided to determine the quarter at which a student can start taking CSE 400 level course by their "smartness" which we calculated from the number of AP Credits that the student has. The logic is that, if they have a great number of AP credits, they are prepared well for college coursework. With 20 AP Credits, the quarter that they can start 400-level turns out to be the 5th quarter (sophomore year). If they have less AP Credits, this number would have them take 400-level courses at the a later quarter (6th or later), which is junior and senior years like any student would. The smarter the student, the earlier the student can start taking these courses.

Constraint (11) states that a student cannot take more than 3 CSE courses or more than 5 courses in total at any quarter. This constraint is not necessary. We only added it to ensure that the student's workload is not too heavy so that our solution is more realistic and reasonable.

4 Solution

We wrote a python script (attached in Appendixes) to generate the input text file for the LP solve program. This solution has the following initial conditions:

- Maximum credit allowed per quarter: 15
- AP credits from high school: 20

Run time of the LP_solve program:
CPU Time for Parsing input: 0.02981s (0.02981s total since program start)
CPU Time for solving: 3.74184s (3.77165s total since program start)
The run time is much faster than we expected. Through some experiments, we found out that the constraint of course prerequisite causes the most complexity (Constraint 6). Without applying constraint 6, the run time is on average about 1.00s.

Value of objective function:		x_8_CSE333	1	z_VLPA4A	1
11.00000000		x_8_CSE428	1	z_QSR4A	1
		x_8_W5C	1	z_QSR4D	1
Actual values of the		x_9_CSE457	1	z_W4D	1
variables:		x_9_CSE454	1	z_VLPA5A	1
y_1	1	x_9_QSR2A	1	z_VLPA5B	1
y_2	1	x_9_VLPA4A	1	z_VLPA5D	1
y_3	1	x_10_CSE458	1	z_QSR5A	1
y_4	1	x_10_VLPA5A	1	z_QSR5B	1
y_5	1	x_10_QSR5A	1	z_QSR5C	1
y_6	1	x_11_CSE459	1	z_QSR5D	1
y_7	1	x_11_QSR2B	1	z_W5C	1
y_8	1	x_11_W3A	1		
y_9	1	x_11_QSR5C	1		
y_10	1	z_CSE142	1		
y_11	1	z_CSE143	1		
x_1_CSE142	1	z_CSE311	1		
x_1_MATH124	1	z_CSE312	1		
x_1_VLPA2A	1	z_CSE331	1		
x_1_QSR4A	1	z_CSE332	1		
x_2_CSE143	1	z_CSE333	1		
x_2_MATH125	1	z_CSE344	1		

Figure 1: Truncated LP solution output file

Figure 1 is a truncated version of the output file from LP showing a part of all the non-zero values. To make the result more readable, we parsed the output file into the Python script and reformatted it into a list as shown in Figure 2.
Our LP input file contains 2875 lines of constraints, we will mostly display 2 lines of each.

```
Student with 20 AP credits
graduated in 11 quarters

Quarter 1 Freshman
 CSE120 cr: 5
 CSE142 cr: 4
MATH124 cr: 5
 VLPA1A cr: 1
Quarter credit: 15
Total credit: 35

Quarter 2 Freshman
 CSE143 cr: 5
MATH125 cr: 5
 VLPA1B cr: 1
 VLPA4B cr: 4
Quarter credit: 15
Total credit: 50

Quarter 3 Sophomore
 CSE154 cr: 5
 CSE351 cr: 4
MATH126 cr: 5
 VLPA1C cr: 1
Quarter credit: 15
Total credit: 65

Quarter 4 Sophomore
 CSE163 cr: 4
 CSE311 cr: 4
 CSE331 cr: 4
MATH307 cr: 3
Quarter credit: 15
Total credit: 80

Quarter 5 Sophomore
 CSE332 cr: 4
 VLPA2A cr: 2
    W4D cr: 4
 VLPA5C cr: 5
Quarter credit: 15
Total credit: 95

Quarter 6 Junior
 CSE440 cr: 5
 VLPA5B cr: 5
    W5D cr: 5
Quarter credit: 15
Total credit: 110

Quarter 7 Junior
 CSE312 cr: 4
 CSE402 cr: 4
 VLPA2B cr: 2
  QSR5D cr: 5
Quarter credit: 15
Total credit: 125

Quarter 8 Junior
 CSE333 cr: 4
 CSE442 cr: 4
 CSE464 cr: 3
  QSR4C cr: 4
Quarter credit: 15
Total credit: 140

Quarter 9 Senior
 CSE401 cr: 4
 CSE457 cr: 4
ENGLxxx cr: 5
 VLPA2D cr: 2
Quarter credit: 15
Total credit: 155

Quarter 10 Senior
 CSE458 cr: 5
 CSE428 cr: 5
  QSR5C cr: 5
Quarter credit: 15
Total credit: 170

Quarter 11 Senior
 CSE441 cr: 5
MATH308 cr: 3
    W2A cr: 2
  QSR5B cr: 5
Quarter credit: 15
Total credit: 185
```

Figure 2: Formatted solution

LP input file

```
min: y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + y_10 + y_11 + y_12;

(constraint1)(12 lines of the following type: ensure that each quarter
must be taken iff at least one credit is taken with some courses)
x_1_CSE120 + . . . + x_1_W5D <= 20 y_1;
:
x_12_CSE120 + . . . + x_12_W5D <= 20 y_12;

(constraint1)(12 lines of the following type:ensure that each quarter
must be taken iff at least one credit is taken with some courses)
y_1 <= x_1_CSE120 + . . . + x_1_W5D;
:
y_12 <= x_12_CSE120 +  . . . + x_12_W5D;

(constraint2)(108 lines of the following type: ensure that whether the course
has been taken or not,for each course)
x_1_CSE120 + x_2_CSE120 + . . .   + x_12_CSE120 <= 20 z_CSE120;
:
x_1_W5D + x_2_W5D +  . . .  + x_12_W5D <= 20 z_W5D;

(constraint2)(108 lines of the following type: ensure that whether the course
has been taken or not, for each course )
```

```
z_CSE120 <= x_1_CSE120 + x_2_CSE120 +      + x_12_CSE120;
:
z_W5D < = x_1_W5D + x_2_W5D +  . . .  + x_12_W5D;

(constraint3)(11 lines of the following type: ensure that lower numbered quarter
has been enrolled before higher numbered quarter)
y_2 <= y_1;
:
y_12 <= y_11;

(constraint4)(1296 lines of the following type: Ensure that each courses are
    offered in all quarters)
x_1_CSE120 <= o_1_CSE120
:
x_12_W5D < = o_12_W5D

(constraint5)(1296 lines of the following type:Ensure that each course can only be
    taken once)
r: x_1_CSE120 <= 1;
:
r: x_12_W5D <= 1;

(constraint6)(880 lines of the following type:ensure that certain course
must be taken after completing the prerequisites courses)
x_1_CSE142 >= x_2_CSE143
:
x_1_AMATH126+x_2_AMATH126+ . . .  + x_11_AMATH126  >= x_12_AMATH352

(constraint7)(12 lines of the following type:Ensure that each quarter has maximum
    of 15 credits)
5 x_1_CSE120 + . . . + 5 x_1_W5D <= 15;
:
5 x_12_CSE120 + . . . + 5 x_12_W5D <= 15;

(constraint8)( 6 lines of the following type:Ensure that each credit type satisfy
with certain graduation credit)
1 z_VLPA1A + . . .  + 5 z_VLPA5D >= 20;
1 z_QSR1A + . . . + 5z_QSR5D >= 5
1 z_W1A + . . .+ 5 z_W5D >= 10;
4 z_CSE401 + . . . + 5 z_CSE460 >= 33;
5 z_MATH124 + . . . + 3 z_MATH308 >= 18;
5 z_CSE120 + . . . + 5 z_W5D >= 160;

(constraint 9)( 12 lines of the following:Ensure that CSE core courses should be
```

```
    taken once)
r: z_CSE142 = 1;
:
r: z_ENGLxxx = 1;

(constraint 10)(160 lines of the following:Ensure that no 400-level CSE courses for
     freshman year,
dependent on 'SMARTNESS = 7 - int(APCREDITS / 10)': more APCREDITS allows to take
    400-levels earlier)
r: x_1_CSE401 = 0;
:
r: x_5_CSE460 = 0;

(constraint 11)(12 lines of the following: Ensure that a student should not
take more than 3 CSE courses each quarter)
x_1_CSE120 + . . . + x_1_CSE460 <= 3;
:
x_12_CSE120 + . . . + x_12_CSE460 <= 3; has

(constraint 11)(12 lines of the following:Ensure that a student should not
take more than 5 courses each quarter )
x_1_CSE120 + . . . + x_1_W5D <= 5;
:
x_12_CSE120 + . . . + x_12_W5D <= 5;

bin x_1_CSE120,  . . . , x_12_W5D, z_CSE120, . . . , z_W5D, y_1, . . . , y_12;
```

5 Commentary on Solution

We are very satisfied with our result in the first solution shown in Figure 2. It concludes that a student who comes to the UW with 20 AP credits can potentially graduate in less than 4 years while strictly fulfilling all requirements for the CSE degree. In addition, we are limiting the maximum number of credits to 15 per quarter, which means that the student can graduate in 11 quarters with ease. In Figure 2 it is shown that by the 3rd quarter at the UW, the student is already considered a sophomore student. This is because we consider the sum of the credits they earned at the UW and the AP Credits to calculate what student's year they are in by credit range. So by the end of the 2nd quarter, the student already has enough credits to be considered a sophomore student.

The decision of the student already having 20 AP credits is to apply this problem in a more realistic condition, because nowadays lots of students at the UW took series like Calculus and Economics during high school. And since they are highly prepared for college coursework, they should not need to take the entire 180 credits in order to graduate. We are especially happy about the decision of implementing the course order and prerequisite constraints and its effect on the result. Once these constraints are applied, the resulting schedule is more similar to an actual student's schedule in 4 years. From Figure 2, we can see that the student has a good balance between science and non-sciences classes for each quarter. Thus, the student won't be taking any upper 400-level before their junior year.

Since our problem is Integer Programming, the idea of binding isn't an interesting topic for most of our constraints. However, there are several adjustable variables that would change the solution. max_q is one of these variables. Lowing the cap on the maximum credit allowed per quarter will increase the number of quarters required to graduate, vice versa. The effect of this change is obvious, the less courses a student take each quarter, the more quarters it's required to graduate. Constraint 11 can also be adjusted to affect the solution because we are currently limiting the maximum number of CSE courses allowed per quarter to 3. With more CSE courses possibly taken each quarter, the student can possibly graduate faster.

Inevitably, there are several flaws in our solution. Even when our schedule solution gives a possible schedule path for the student, we can't imagine any CS student taking CSE 120 course as their first CS course at the UW and taking MATH 308 in their senior year. In reality, people would consider the content and the importance of each course before registration. Since the UW is a big public university, it is also entirely possible that our student would not be able to register for the classes as listed in his freshman or sophomore years' schedule unless he's in a FIG (First-year Interest Group), which classes reserve seats for the particular FIGs. In reality, the student could also fail a class and has to take it the subsequent quarter, our program does not account for this possibility but it would be a possible use for the paper we mentioned before by Guleria and Lal (2019). We can observe that

perfect optimization does not correlate with real life situations. Our schedule solution makes a lot of assumption on this particular student. But since we are practicing Integer Programming, it is fine to not taking into account this aspect of the 'human' student.

We would like to perform a series of experiments with different initial conditions to check how these two conditions would affect the result of the optimization. The two variables (AP credits awarded and maximum credits per quarter) will be tested in the range as listed below since we still want our solution to be reasonable. The results will be given in a table.

- max_q, Maximum credit allowed per quarter: 12-18
- AP credits from high school: 0-60

max_q **AP Credit**	**0**	**20**	**40**	**60**
12	15(infeasible)	14(infeasible)	12	10
14	13(infeasible)	12	10	9
16	12	10	9	8
18	10	9	8	7

Note that the value inside the chart is the minimum number of quarters required to graduate based on the given initial condition. Figure3 showed that student required minimum 7 quarters to graduate which was same as the value inside the chart. For 3 entries, the result is infeasible, so we adjusted the maximum number of quarters allowed to check for a possible result. The interesting part is that under every condition, the program is always able to find a solution at the mathematical minimum number of quarters (dividing total credits by max credit per quarter). It is also remarkable to observe from our results that, even if students don't have any AP Credit, they can still graduate within 10 quarters if they were to take 18 credits for each of the 10 quarters. Although the program claims a feasible solution with this setting, this kind of schedule is probably not likely to be seen in real life.

```
Student with 60 AP credits
graduated in 7 quarters

Quarter 1 Sophomore
 CSE142 cr: 4
 CSE131 cr: 4
MATH124 cr: 5
ENGLxxx cr: 5
Quarter credit: 18
Total credit: 78

Quarter 2 Sophomore
 CSE120 cr: 5
 CSE143 cr: 5
 CSE495 cr: 3
MATH125 cr: 5
Quarter credit: 18
Total credit: 96

Quarter 3 Junior
 CSE331 cr: 4
 CSE464 cr: 3
MATH126 cr: 5
 VLPA2A cr: 2
    W4A cr: 4
Quarter credit: 18
Total credit: 114

Quarter 4 Junior
 CSE311 cr: 4
 CSE351 cr: 4
 CSE456 cr: 4
    W1B cr: 1
 VLPA5A cr: 5
Quarter credit: 18
Total credit: 132

Quarter 5 Junior
 CSE332 cr: 4
    W4C cr: 4
 VLPA5D cr: 5
  QSR5D cr: 5
Quarter credit: 18
Total credit: 150

Quarter 6 Senior
 CSE312 cr: 4
 CSE401 cr: 4
 CSE440 cr: 5
 VLPA5B cr: 5
Quarter credit: 18
Total credit: 168

Quarter 7 Senior
 CSE402 cr: 4
 CSE441 cr: 5
 CSE454 cr: 5
MATH308 cr: 3
    W1A cr: 1
Quarter credit: 18
Total credit: 186
```

Figure 3: LP solution with 60 AP credits and max of 18 credits per quarter

6 Variation

Variation 1: A change in our objective function

The first variation we came up with was to change our objective from minimizing the number of quarters to minimizing or maximizing the number of CSE courses a CSE students takes during 4 years in UW. Consequently, the first update we made to our model was changing our objective function to the following:

$$\text{minimize / maximize: } \sum_c z_c k_{crtp,c} \text{ where } crtp = \text{"CSE"}$$

Maximizing or minimizing the above objective function correspond to maximizing or minimizing the total number of CSE courses. Since we are no longer accounting for the minimized number of quarters to graduate, $\sum_{q=1}^{12} y_q$, is no longer being constrained. This way Constraint (3) becomes unnecessary and is discarded for this variation. We do not need to consider the order of the quarters since we already fixed the quarters to 12. So for this variation, we fixed the number of quarters to graduate to 12 explicitly, that is,

$$y_q = 1, \;\; q = 1, ..., 12.$$

The normal number of total CSE courses to graduate in 4 years was around 20. Comparing to the normal result, we get Figure 4 of a schedule with the minimized number of CSE courses. By limiting the credits per quarter to 15 credits and the student having 20 AP credits, the student had 189 total credits and takes only 15 CSE courses. Contrast to Figure 4, Figure 5 shows the maximum number of CSE courses possible in 4 years, which turns to be 33. Not only were there differences

```
Student with 20 AP credits
graduated in 12 quarters

Quarter 1 Freshman
 CSE142 cr: 4
MATH124 cr: 5
    W4B cr: 4
Quarter credit: 13
Total credit: 33

Quarter 2 Freshman
MATH125 cr: 5
 VLPA4D cr: 4
    W4A cr: 4
Quarter credit: 13
Total credit: 46

Quarter 3 Sophomore
 CSE143 cr: 5
MATH126 cr: 5
  QSR4B cr: 4
Quarter credit: 14
Total credit: 60

Quarter 4 Sophomore
 CSE311 cr: 4
 CSE331 cr: 4
 CSE351 cr: 4
MATH307 cr: 3
Quarter credit: 15
Total credit: 75

Quarter 5 Sophomore
 CSE332 cr: 4
 CSE333 cr: 4
MATH308 cr: 3
    W4C cr: 4
Quarter credit: 15
Total credit: 90

Quarter 6 Junior
 CSE440 cr: 5
  QSR4C cr: 4
  QSR4D cr: 4
Quarter credit: 13
Total credit: 103

Quarter 7 Junior
 CSE312 cr: 4
 CSE457 cr: 4
 VLPA3C cr: 3
    W4D cr: 4
Quarter credit: 15
Total credit: 118

Quarter 8 Junior
 CSE428 cr: 5
 VLPA3A cr: 3
 VLPA3B cr: 3
  QSR4A cr: 4
Quarter credit: 15
Total credit: 133

Quarter 9 Junior
STAT391 cr: 4
 VLPA4B cr: 4
    W5D cr: 5
Quarter credit: 13
Total credit: 146

Quarter 10 Senior
 CSE458 cr: 5
 CSE454 cr: 5
STAT390 cr: 4
Quarter credit: 14
Total credit: 160

Quarter 11 Senior
 CSE459 cr: 5
 CSE441 cr: 5
 VLPA4A cr: 4
Quarter credit: 14
Total credit: 174

Quarter 12 Senior
AMATH351 cr: 3
AMATH352 cr: 3
ENGLxxx cr: 5
 VLPA4C cr: 4
Quarter credit: 15
Total credit: 189
Student took a total of 15 CSE
courses.
```

Figure 4: Solution of minimizing the number of CSE courses

between the number of CSE courses, but there were also differences between total credits earned and credit per quarter. Figure 5 showed a very interesting result. The student had earned a total of 200 credits before graduation. Since we were no longer minimizing the number of quarters required to graduate, the program didn't care about how many quarters it will take, instead, it used up every single quarter available. In reality, there were also many students over 180 credit when graduated.

```
Student with 20 AP credits
graduated in 12 quarters

Quarter 1 Freshman
 CSE142 cr: 4
 CSE160 cr: 4
MATH124 cr: 5
 VLPA1A cr: 1
    W1A cr: 1
Quarter credit: 15
Total credit: 35

Quarter 2 Freshman
 CSE143 cr: 5
 CSE163 cr: 4
MATH125 cr: 5
    W1C cr: 1
Quarter credit: 15
Total credit: 50

Quarter 3 Sophomore
 CSE340 cr: 4
 CSE341 cr: 4
 CSE351 cr: 4
    W3A cr: 3
Quarter credit: 15
Total credit: 65

Quarter 4 Sophomore
 CSE154 cr: 5
 CSE331 cr: 4
 CSE391 cr: 1
MATH126 cr: 5
Quarter credit: 15
Total credit: 80

Quarter 5 Sophomore
 CSE311 cr: 4
 CSE333 cr: 4
    W2A cr: 2
 VLPA5A cr: 5
Quarter credit: 15
Total credit: 95

Quarter 6 Junior
 CSE131 cr: 4
 CSE312 cr: 4
 CSE332 cr: 4
 VLPA3B cr: 3
Quarter credit: 15
Total credit: 110

Quarter 7 Junior
 CSE427 cr: 3
 CSE431 cr: 3
 CSE447 cr: 4
ENGLxxx cr: 5
Quarter credit: 15
Total credit: 125

Quarter 8 Junior
 CSE403 cr: 4
 CSE446 cr: 4
 CSE455 cr: 4
MATH308 cr: 3
Quarter credit: 15
Total credit: 140

Quarter 9 Senior
 CSE344 cr: 4
 CSE421 cr: 3
 CSE457 cr: 4
 VLPA4B cr: 4
Quarter credit: 15
Total credit: 155

Quarter 10 Senior
 CSE402 cr: 4
 CSE444 cr: 4
 CSE451 cr: 4
 VLPA3C cr: 3
Quarter credit: 15
Total credit: 170

Quarter 11 Senior
 CSE401 cr: 4
 CSE452 cr: 4
 CSE456 cr: 4
    W3C cr: 3
Quarter credit: 15
Total credit: 185

Quarter 12 Senior
 CSE440 cr: 5
 CSE442 cr: 4
 CSE458 cr: 5
 VLPA1C cr: 1
Quarter credit: 15
Total credit: 200
Student took a total of 33 CSE
courses.
```

Figure 5: Solution of maximizing the number of CSE courses

Variation 2: Implementing Real data for Course Offering Information

In our original solution, we were considering the ideal condition that every course is offered every quarter in order to give LP solver more feasible region. In this variation, we now consider the effect of course offering onto our model.We made a new Python script that craws through CSE Time Schedules and fetched all the course offering information into our program. On the website, each course has its assigned professor's name listed in the chart. If the name appears in an entries, then we consider that course is offered in that quarter. Since there are lots of minor changes on the website each over, we used the official time schedule for 2019-2020 as a template for all 12 quarters in our model. For example, the offering in autumn quarter will be the information in quarter 1, 4, 7, 10. Previously, the constraint of course offering (Constraint 4) has all O_{qc} equal to 1, but now we can make the program read the scraped data and assign value for O_{qc} accordingly.

After the implementation, we ran the program with the original objective function (minimize the number of quarters used), and here are the results between the new solution and the original solution with the same initial condition:

```
Student with 20 AP credits
graduated in 11 quarters

Quarter 1 Freshman
 CSE142 cr: 4
MATH124 cr: 5
 VLPA2C cr: 2
  QSR3C cr: 3
Quarter credit: 14
Total credit: 34

Quarter 2 Freshman
 CSE143 cr: 5
MATH125 cr: 5
  QSR5D cr: 5
Quarter credit: 15
Total credit: 49

Quarter 3 Sophomore
MATH126 cr: 5
MATH307 cr: 3
 VLPA3C cr: 3
 VLPA4D cr: 4
Quarter credit: 15
Total credit: 64

Quarter 4 Sophomore
 CSE311 cr: 4
 CSE331 cr: 4
 CSE351 cr: 4
 VLPA3B cr: 3
Quarter credit: 15
Total credit: 79

Quarter 5 Sophomore
 CSE160 cr: 4
 CSE312 cr: 4
 CSE332 cr: 4
    W3A cr: 3
Quarter credit: 15
Total credit: 94

Quarter 6 Junior
 VLPA1A cr: 1
  QSR4B cr: 4
    W5B cr: 5
    W5C cr: 5
Quarter credit: 15
Total credit: 109

Quarter 7 Junior
 CSE333 cr: 4
 CSE474 cr: 4
    W3B cr: 3
 VLPA4C cr: 4
Quarter credit: 15
Total credit: 124

Quarter 8 Junior
 CSE163 cr: 4
 CSE402 cr: 4
 CSE442 cr: 4
MATH308 cr: 3
Quarter credit: 15
Total credit: 139

Quarter 9 Senior
 VLPA5C cr: 5
    W5A cr: 5
    W5D cr: 5
Quarter credit: 15
Total credit: 154

Quarter 10 Senior
 CSE455 cr: 4
 CSE478 cr: 4
ENGLxxx cr: 5
Quarter credit: 13
Total credit: 167

Quarter 11 Senior
 CSE440 cr: 5
 CSE446 cr: 4
 CSE461 cr: 4
Quarter credit: 13
Total credit: 180

Student took a total of 18 CSE
courses.
```

Figure 6: Solution considering realistic course offering

After running the program, we first noticed that the run time had increased slightly by roughly 0.1 second: CPU Time for solving: 3.82246s (3.85256s total since program start)

Comparing these 2 solutions (figure 6,7), we first noticed that the total number of quarters required to graduate didn't change. Looking into the course offering schedule, we found out that for

```
Student with 20 AP credits
graduated in 11 quarters

Quarter 1 Freshman
 CSE120 cr: 5
 CSE142 cr: 4
MATH124 cr: 5
 VLPA1A cr: 1
Quarter credit: 15
Total credit: 35

Quarter 2 Freshman
 CSE143 cr: 5
MATH125 cr: 5
 VLPA1B cr: 1
 VLPA4B cr: 4
Quarter credit: 15
Total credit: 50

Quarter 3 Sophomore
 CSE154 cr: 5
 CSE351 cr: 4
MATH126 cr: 5
 VLPA1C cr: 1
Quarter credit: 15
Total credit: 65

Quarter 4 Sophomore
 CSE163 cr: 4
 CSE311 cr: 4
 CSE331 cr: 4
MATH307 cr: 3
Quarter credit: 15
Total credit: 80

Quarter 5 Sophomore
 CSE332 cr: 4
 VLPA2A cr: 2
    W4D cr: 4
 VLPA5C cr: 5
Quarter credit: 15
Total credit: 95

Quarter 6 Junior
 CSE440 cr: 5
 VLPA5B cr: 5
    W5D cr: 5
Quarter credit: 15
Total credit: 110

Quarter 7 Junior
 CSE312 cr: 4
 CSE402 cr: 4
 VLPA2B cr: 2
  QSR5D cr: 5
Quarter credit: 15
Total credit: 125

Quarter 8 Junior
 CSE333 cr: 4
 CSE442 cr: 4
 CSE464 cr: 3
  QSR4C cr: 4
Quarter credit: 15
Total credit: 140

Quarter 9 Senior
 CSE401 cr: 4
 CSE457 cr: 4
ENGLxxx cr: 5
 VLPA2D cr: 2
Quarter credit: 15
Total credit: 155

Quarter 10 Senior
 CSE458 cr: 5
 CSE428 cr: 5
  QSR5C cr: 5
Quarter credit: 15
Total credit: 170

Quarter 11 Senior
 CSE441 cr: 5
MATH308 cr: 3
    W2A cr: 2
  QSR5B cr: 5
Quarter credit: 15
Total credit: 185
```

Figure 7: The original solution

the most of the core courses required to graduate, the department always offer them in every quarter. Hence this should allow any student to graduate without a problem in registration. However, we also noticed that in the new version of solution, the student would no longer take easy and weird courses like CSE 120 and CSE 160. The reason is that these courses are not offered very often as they are not meant for CS students to take. Another difference we noticed is that in the new solution, there are 18 CSE courses in the schedule while there are 20 in the original one. This difference agrees with our assumption that with the limitation applied to course offering, the student would take less CSE courses in total. As the result, we think that the schedule appears even more realistic with this improvement.

7 Conclusion

This paper shows one of the many ways how an optimized student plan can be derived from Integer Programming. We showed that, in an ideal situation, a student can easily graduate in less than 4 years with a BS degree in Computer Science. As we reached our solution, we observed that this method can be implemented in various other student scheduling situations, mostly depends on each individual's thoughts.

Two most important outcomes that we obtain from our study is the variation of approaches and the compromise within the model. Since the topic of scheduling, especially in long-term, has a lot of fixed ideals on the individual and the environment. The program that resolves the problem must be dynamically changing to constantly adapt to new situations. Moreover, there may be various changes to the each student's performances and mental health that are not being considered when we are using Integer Programming to come up with all the constraints.

Overall, with well designed constraints, we think that our solution had shown us a nice broad picture of the flexibility and feasibility of the schedule planning in the CSE department of UW. The program produced lots of fruitful results for us to learn and to analyze.

8 Bibliography

Appleby, J. S. “Techniques for Producing School Timetables on a Computer and Their Application to Other Scheduling Problems.” The Computer Journal 3, no. 4 (January 1961): 237–245.

“COLLEGE OF ENGINEERING COMPUTER SCIENCE AND ENGINEERING COMPUTER SCIENCE ENGINEERING.” COMPUTER SCIENCE ENGINEERING. Accessed February 7, 2020. https://www.washington.edu/students/crscat/cse.html.

CSE Teaching Schedule 2019-2020. Accessed February 7, 2020. https://s3-us-west-2.amazonaws.com/www-cse-public/education/time-sched/teaching2019-2020.html.

“Degree Requirements.” Degree Requirements | Paul G. Allen School of Computer Science Engineering. Accessed February 7, 2020. https://www.cs.washington.edu/academics/ugrad/courses/requirements.

Feldman, R. “Optimization Algorithms for Student Scheduling via Constraint Satisfiability.” The Computer Journal 33, no. 4 (January 1990): 356–64. https://doi.org/10.1093/comjnl/33.4.356.

Guo, Jiayi, David R. Morrison, Sheldon H. Jacobson, and Janet A. Jokela. “Complexity Results for the Basic Residency Scheduling Problem.” Journal of Scheduling 17, no. 3 (2013): 211–23. https://doi.org/10.1007/s10951-013-0362-9.

Guleria, Sanjeev K., and Arvind K. Lal. “Interactive Decision Support System for Planned Student Scheduling Using Same-Time Scheduling in Timetable.” Modern Physics Letters B 33, no. 13 (October 2019): 1950161. https://doi.org/10.1142/s0217984919501616.

https://doi.org/10.1093/comjnl/3.4.237.Feldman, R. “Optimization Algorithms for Student Scheduling via Constraint Satisfiability.” The Computer Journal 33, no. 4 (January 1990): 356–64. https://doi.org/10.1093/comjnl/33.4.356.

Salazar, A., and R. V. Oakford. “A Graph Formulation of a School Scheduling Algorithm.” Communications of the ACM 17, no. 12 (January 1974): 696–98.https://doi.org/10.1145/361604.361625.

9 Appendixes

```
"""
This is the python script that generates the input
for the LP solver for the purpose of generating the
input for the schedule solver.
"""
import os

N = 12 # number of total quarters
MAXCREDIT = 15 # maximum credit per quarter
APCREDITS = 20 # credit student bring to college from highscool
GRADCREDIT = 180
SMARTNESS = 7 - int(APCREDITS / 10)
allQuarters = range(1, N+1) # adjusted range for Python
fullCourseList = [] # list of course names
coursePrereqMap = {} # course name => list of course names
courseCrtMap = {} # course name => integer
courseCrtTypeMap = {} # course name => credit type
courseOfferingMap = {} # course name => list of booleans
creditTypes = ["VLPA", "QSR", "W", "CSELECTIVE", "MELECTIVE"] # all credits types
creditRequirement = {} # credit requirement of each credit type

def main():
    output = open("scheSolve.txt","w+")
    loadAllCourses()
    loadGradRequirement()
    objectiveF(output)
    constraint1(output)
    constraint2(output)
    constraint3(output)
    constraint4(output)
    constraint5(output)
    constraint6(output)
    constraint7(output)
    constraint8(output)
    constraint9(output)
    constraint10(output)
    constraint11(output)
    loadVaribles(output)
    output.close()
    # This command only works if you have lp_solve installed in the terminal
    print("Solving...")
    os.system("lp_solve -s scheSolve.txt > scheSolution.txt")
```

```
    getSolution()

# Add unnamed VLPA, QSR, W credits
def loadAllCourses():
    loadCoreCourses()
    for cr in range(1,6):
        for type in creditTypes[:-2]:
            for suffix in ["A", "B", "C", "D"]:
                name = type + str(cr) + suffix
                fullCourseList.append(name)
                courseCrtMap[name] = cr
                courseCrtTypeMap[name] = type
    # Currently we assume that all courses are offered every quarter
    for course in fullCourseList:
        courseOfferingMap[course] = [True, True, True]

# Load all CSE Core courses we consider
# Prerequisite map is not loaded yet
def loadCoreCourses():
    fileInput = open("courseCrtInfo.txt", "r")
    lines = fileInput.readlines()
    for line in lines:
        line = line.split("|")
        fullCourseList.append(line[0])
        if (len(line) > 2):
            courseCrtTypeMap[line[0]] = line[2][:-1]
            courseCrtMap[line[0]] = line[1]
        else:
            courseCrtMap[line[0]] = line[1][:-1]
    fileInput.close()

# Load the credit requirement for graduation
def loadGradRequirement():
    creditRequirement["VLPA"] = 20;
    creditRequirement["QSR"] = 5;
    creditRequirement["W"] = 10;
    creditRequirement["CSELECTIVE"] = 33;
    creditRequirement["MELECTIVE"] = 18;

# Objective Function
def objectiveF(f):
    objFun = ""
    for q in allQuarters:
        objFun += "y_%d + " % q
```

```
87      f.write("min: " + objFun[:-3] + ";\n")
88
89  # Constraint 1
90  # Each quarter must be taken if and only if at least one
91  # credit is taken with some courses
92  def constraint1(f):
93      for q in allQuarters:
94          line = ""
95          for c in fullCourseList:
96              line += "x_%d_%s + " % (q, c)
97          f.write(line[:-3] + " <= 20 y_%d;\n" % q)
98          f.write("y_%d <= " % q + line[:-3] + ";\n")
99
100 # Constraint 2
101 # For each course, match with a variable indicating
102 # whether the course has been taken or not
103 def constraint2(f):
104     for c in fullCourseList:
105         line = ""
106         for q in allQuarters:
107             line += "x_%d_%s + " % (q, c)
108         f.write(line[:-3] + " <= 20 z_%s;\n" % c)
109         f.write("z_%s <= " % c + line[:-3] + ";\n")
110
111 # Constraint 3
112 # Each quarter must be used in order
113 def constraint3(f):
114     for q in allQuarters[1:]:
115         f.write("y_%d <= y_%d;\n" % (q, q-1))
116
117 # Constraint 4
118 # For each course, make sure that each course can only
119 # be taken if and only if it's offered in that particular
120 # quarter
121 def constraint4(f):
122     for c in fullCourseList:
123         for q in allQuarters:
124             if (courseOfferingMap[c][(q-1) % 3]):
125                 f.write("r: x_%d_%s <= %d;\n" % (q, c, 1))
126             else:
127                 f.write("r: x_%d_%s <= %d;\n" % (q, c, 0))
128
129 # Constraint 5
130 # For each course, make sure that each course can only
```

```
# be taken once (assuming student does not fail any courses)
def constraint5(f):
    for c in fullCourseList:
        line = ""
        for q in allQuarters:
            line += "x_%d_%s + " % (q, c)
        f.write(line[:-3] + " <= %d;\n" % 1)

# Constraint 6
# Course Prerequisite, done by loading the prerequisites
# data file
def constraint6(f):
    fileInput = open("prereqLPInput.txt", "r")
    lines = fileInput.readlines()
    fileInput.close()
    for info in lines:
        info = info.split("|")
        course = info[0]
        prereqs = info[1][:-1]
        # Any course with a prereq cannot be taken in the first quarter
        f.write("r: x_%d_%s <= %d" % (1, course, 0) + ";\n")
        if "+" in prereqs:
            prereqs = prereqs.split("+")
            for q2 in allQuarters[1:]:
                line = "x_%d_%s <= " % (q2, course)
                for c in prereqs:
                    for q1 in allQuarters[:q2-1]:
                        line += "x_%d_%s + " % (q1, c)
                f.write(line[:-3] + ";\n")
        else:
            prereqs = prereqs.split(";")
            for q2 in allQuarters[1:]:
                for c in prereqs:
                    line = "x_%d_%s <= " % (q2, course)
                    for q1 in allQuarters[:q2-1]:
                        line += "x_%d_%s + " % (q1, c)
                    f.write(line[:-3] + ";\n")

# Constraint 7
# For each quarter, all courses cannot exceed the credit limit
def constraint7(f):
    for q in allQuarters:
        line = ""
        for c in fullCourseList:
```

```
            line += "%s x_%d_%s + " % (courseCrtMap[c], q, c)
        f.write(line[:-3] + " <= %d;\n" % MAXCREDIT)

# Constraint 8
# Each credit type requirement for graduation should be satisfied
# To do so we count credits from every course
def constraint8(f):
    for crtType in creditRequirement:
        line = ""
        for c in fullCourseList:
            if (c in courseCrtTypeMap and courseCrtTypeMap[c] == crtType):
                line += "%s z_%s + " % (courseCrtMap[c], c)
        f.write(line[:-3] + " >= %d;\n" % creditRequirement[crtType])
    line = ""
    for c in fullCourseList:
        line += "%s z_%s + " % (courseCrtMap[c], c)
    f.write(line[:-3] + " >= %d;\n" % (GRADCREDIT-APCREDITS))

# Constraint 9
# CSE MAJOR REQUIREMENT
# For the CSE major, you must take core courses
# TODO
def constraint9(f):
    fileInput = open("coreRequirement.txt", "r")
    lines = fileInput.readlines()
    for line in lines:
        line = line[:-1]
        f.write("r: z_%s" % line + " = %d;\n" % 1)
    fileInput.close()

# Constraint 10
# A student is not expected to take 400 level classes in freshman year
# unless his smartness is really good
def constraint10(f):
    for q in allQuarters[:SMARTNESS]:
        line = ""
        for c in fullCourseList:
            if (c.startswith("CSE4")):
                f.write("r: x_%d_%s = 0;\n" % (q, c))

# Constraint 11
# A student should not take more than 3 CSE courses each quarter
# A student should not take more than 5 courses in total each quarter
def constraint11(f):
```

```
    for q in allQuarters:
        line1 = ""
        line2 = ""
        for c in fullCourseList:
            line2 += "x_%d_%s + " % (q, c)
            if (c.startswith("CSE")):
                line1 += "x_%d_%s + " % (q, c)
        f.write(line1[:-3] + " <= 3;\n")
        f.write(line2[:-3] + " <= 5;\n")

# Load all variables in LP format
def loadVaribles(f):
    f.write("bin ")
    for q in allQuarters:
        for c in fullCourseList:
            f.write("x_%d_%s, " % (q, c))
    for c in fullCourseList:
        f.write("z_%s, " % c)
    str = ""
    for q in allQuarters:
        str += "y_%d, " % q
    f.write(str[:-2] + ";\n")

# This function only works if you have lp_solve installed in the terminal
def getSolution():
    usedQuarters = []
    classSchedule = []
    takenClasses = []
    f = open("scheSolution.txt","r")
    lines = f.readlines()
    if (len(lines) == 1):
        print("Problem is infeasible.")
        return
    for q in allQuarters:
        classSchedule.append([])
    for line in lines:
        line = line.split()
        if(len(line) > 1 and line[1] == "1"):
            if(line[0].startswith("y")):
                usedQuarters.append(line[0].split("_")[1])
            elif(line[0].startswith("x")):
                data = line[0].split("_")
                classSchedule[int(data[1]) - 1].append(data[2])
            elif(line[0].startswith("z")):
```

```
                takenClasses.append(line[0].split("_")[1])
    f.close()
    print("\nStudent with %d AP credits graduated in %d quarters" % (APCREDITS, len
    (usedQuarters)))
    totalCredit = APCREDITS
    for q in allQuarters[:len(usedQuarters)]:
        currClass = ""
        if totalCredit < 45:
            currClass = "Freshman"
        elif totalCredit < 90:
            currClass = "Sophomore"
        elif totalCredit < 135:
            currClass = "Junior"
        else:
            currClass = "Senior"
        print("\nQuarter", q, currClass)
        credit = 0
        for c in classSchedule[q-1]:
            credit += int(courseCrtMap[c])
            print("%7s cr:" % c, courseCrtMap[c])
            #print("%7s" % c)
        totalCredit += credit
        print("Quarter credit: %d" % credit)
        print("Total credit: %d" % totalCredit)

if __name__ == "__main__":
    main()
```

```
MATH124
MATH125
CSE142
MATH126
MATH308
CSE143
ENGLxxx
CSE312
CSE311
CSE331
CSE332
CSE351
```

Figure 8: Core requirement text file

```
CSE120
CSE142
CSE143
CSE154
CSE160
CSE163
CSE131
CSE311
CSE312
CSE331
CSE332
CSE333
CSE340
CSE341
CSE344
CSE351
CSE369
CSE391
CSE401
CSE402
CSE403
CSE421
CSE427
CSE431
CSE440
CSE442
CSE444
CSE446
CSE447
CSE451
CSE452
CSE455
CSE456
CSE457
CSE458
CSE459
CSE461
CSE464
CSE473
CSE474
CSE478
CSE484
CSE486
CSE490
CSE492
CSE495
CSE428
CSE441
CSE454
CSE460
```

Figure 9: All CSE courses in text file

```
CSE120|5|QSR
CSE142|4|QSR
CSE143|5|QSR
CSE154|5|QSR
CSE160|4|QSR
CSE163|4
CSE131|4|VLPA
CSE311|4|QSR
CSE312|4|QSR
CSE331|4
CSE332|4
CSE333|4
CSE340|4
CSE341|4
CSE344|4
CSE351|4
CSE369|3
CSE391|1
CSE401|4|CSELECTIVE
CSE402|4|CSELECTIVE
CSE403|4|CSELECTIVE
CSE421|3|CSELECTIVE
CSE427|3|CSELECTIVE
CSE431|3|CSELECTIVE
CSE440|5|CSELECTIVE
CSE442|4|CSELECTIVE
CSE444|4|CSELECTIVE
CSE446|4|CSELECTIVE
CSE447|4|CSELECTIVE
CSE451|4|CSELECTIVE
CSE452|4|CSELECTIVE
CSE455|4|CSELECTIVE
CSE456|4|CSELECTIVE
CSE457|4|CSELECTIVE
CSE458|5|CSELECTIVE
CSE459|5|CSELECTIVE
CSE461|4|CSELECTIVE
CSE464|3|CSELECTIVE
CSE473|3|CSELECTIVE
CSE474|4|CSELECTIVE
CSE478|4|CSELECTIVE
CSE484|4|CSELECTIVE
CSE486|3|CSELECTIVE
CSE490|3|CSELECTIVE
CSE492|3|CSELECTIVE
CSE495|3|CSELECTIVE
CSE428|5|CSELECTIVE
CSE441|5|CSELECTIVE
CSE454|5|CSELECTIVE
CSE460|5|CSELECTIVE
```

```
MATH124|5|MELECTIVE
MATH125|5|MELECTIVE
MATH126|5|MELECTIVE
MATH307|3|MELECTIVE
MATH308|3|MELECTIVE
STAT390|4
STAT391|4
AMATH351|3
AMATH352|3
ENGLxxx|5
```

Figure 10: Credit for all CSE courses in text file

```
CSE143|CSE142
CSE312|CSE311
CSE331|CSE143
CSE332|CSE311
CSE333|CSE351
CSE340|CSE143
CSE341|CSE143
CSE344|CSE311
CSE351|CSE143
CSE369|CSE311
CSE391|CSE143
CSE401|CSE332;CSE351
CSE403|CSE331;CSE332
CSE421|CSE312;CSE332
CSE427|CSE312;CSE332
CSE431|CSE312
CSE440|CSE332
CSE442|CSE332
CSE444|CSE332;CSE344
CSE455|CSE333;CSE332
CSE457|CSE333;CSE332
CSE458|CSE457
CSE459|CSE458
CSE473|CSE332
CSE474|CSE143
CSE484|CSE332;CSE351
CSE441|CSE440
CSE460|CSE458;CSE459
CSE154|CSE142+CSE143+CSE160
CSE163|CSE142+CSE143+CSE160
CSE311|MATH126
CSE311|CSE143
CSE402|CSE332;CSE351
CSE446|CSE332
CSE446|STAT390+STAT391+CSE312
CSE447|CSE312;CSE332
CSE451|CSE351;CSE332;CSE333
CSE452|CSE333;CSE332
CSE461|CSE332
CSE461|CSE303+CSE333
CSE478|CSE332
CSE486|MATH307+AMATH351
CSE486|MATH308+AMATH352+CSE311
CSE428|CSE312;CSE331;CSE332
CSE454|CSE332;CSE351
CSE454|CSE331
MATH125|MATH124
MATH126|MATH125
MATH308|MATH126
STAT390|MATH126
STAT391|CSE312
AMATH351|MATH125
AMATH352|MATH126
```

Figure 11: Prerequisite for CSE courses in text file

```
CSE120|False|True|False
CSE142|True|True|True
CSE143|True|True|True
CSE154|True|True|True
CSE160|False|True|False
CSE163|False|True|True
CSE131|False|False|True
CSE311|True|True|True
CSE312|True|True|True
CSE331|True|True|True
CSE332|True|True|True
CSE333|True|True|True
CSE340|False|True|True
CSE341|True|True|True
CSE344|True|True|True
CSE351|True|True|True
CSE369|False|True|True
CSE391|True|True|True
CSE401|True|False|True
CSE402|False|True|False
CSE403|False|False|True
CSE421|True|True|True
CSE427|False|True|False
CSE431|True|False|False
CSE440|True|True|False
CSE442|False|True|False
CSE444|False|True|True
CSE446|True|True|True
CSE447|False|True|False
CSE451|True|True|True
CSE452|False|True|True
CSE455|True|False|True
CSE456|False|False|False
CSE457|False|False|True
CSE458|True|False|False
CSE459|True|False|False
CSE461|True|True|True
CSE464|False|False|True
CSE473|True|True|True
CSE474|True|True|True
CSE478|True|True|False
CSE484|True|False|True
CSE486|True|False|False
CSE490|True|True|True
CSE492|False|True|False
CSE495|False|False|False
CSE428|False|False|False
CSE441|False|False|True
CSE454|False|False|False
CSE460|False|True|False
```

Figure 12: Course offering info on quarter: AU|WI|SP(True means offering